UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI SS. MM. FF.

CORSO DI LAUREA IN MATEMATICA

TESI DI LAUREA

Aggregazione non manipolabile di preferenze in sistemi multi-agente.

Relatore: Ch.ma Prof.ssa Francesca Rossi

Laureanda: Luisa Gonzato

A.A. 2003-2004

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

In questa tesi viene affrontato il problema di progettare dei risolutori per problemi vincolati multi-agente, corrispondenti ad alcuni criteri di non manipolabilità.

Un problema vincolato (*Constraint Satisfaction Problem -CSP-*, come viene denominato in [1, 2]), consiste in un insieme V di variabili aventi ciascuna un insieme D di valori ammissibili, e da un insieme C di vincoli sulle variabili. Ogni vincolo specifica quali assegnazioni siano permesse alle incognite coinvolte. Determinarne una soluzione significa trovare un assegnamento di valori a tutte le variabili, che rispetti i domini e soddisfi tutte le limitazioni imposte.
Poichè l'insieme dei vincoli restringe il dominio delle variabili, può succedere che in alcuni casi non vi sia alcuna soluzione al problema considerato. È conveniente allora considerare una classe più flessibile di vincoli, detti "Vincoli Soft" ([1, 2]).
Il formalismo dei vincoli Soft si basa sulla possibilità di associare un grado di preferenza alle assegnazioni, massima per quei valori che soddisfano i vincoli, ma non necesseriamente nulla per quelli che non li rispettano. In questo modo ogni vincolo può essere soddisfatto od eluso; una soluzione al problema consisterà nelle assegnazioni che rendono massimo il grado di preferenza totale.
Quando si devono gestire le valutazioni riportate da più di un agente, ogni vincolo può essere giudicato in modo diverso da chi lo sta considerando. Se si vuole trovare una soluzione comune ad un gruppo di persone, quindi, si deve poter combinare tra loro le valutazioni, in modo da massimizzare la soddisfazione di tutti i partecipanti.
Si suppone, inoltre, che gli agenti partecipanti al sistema riportino le proprie valutazioni seguendo il personale interesse, e che siano in grado di dedurre, con semplici considerazioni, quale sia la scelta per loro più conveniente. Sorge

allora il problema di garantire che la soluzione prodotta sia veramente quella ottimale, che tutti gli agenti, cioè, abbiano riportato le loro valutazioni in modo sincero, senza mentire per aumentare il guadagno personale.
A partire da queste ipotesi, il nostro lavoro si è concentrato sulla realizzazione di meccanismi che consentano di risolvere un problema di assegnazione, massimizzando l'utilità degli agenti e garantendo al tempo stesso la non manipolabilità dei risultati da parte degli agenti. Abbiamo a tale scopo studiato un metodo di aggregazione di preferenze introdotto di recente [6] nell'ambito dell'intelligenza artificiale, cercando un modo per adattarlo al nostro problema: dato un insieme di variabili, un gruppo di agenti riporta le proprie valutazioni sulle assegnazioni di valori che possono essere fatte a coppie di tali variabili, allo scopo di produrre un'assegnazione completa che soddisfi tutti gli agenti. La progettazione del meccanismo di aggregazione di preferenze avviene prima di conoscere le dichiarazioni dei partecipanti al sistema: si suppone che un coordinatore conosca a priori le probabilità con cui ogni agente riporterà ciascun grado di preferenza, e l'utilità ad esso legata. Il meccanismo proposto Conitzer e Sandholm in [6], però, risulta inefficente per quei particolari problemi in cui le valutazioni devono essere riportate su un gran numero di oggetti. Abbiamo, dunque, adattato il modello AMD nel seguente modo: nonostante i risultati siano assegnazioni complete di valori a tutte le variabili del problema, gli agenti partecipanti al meccanismo esprimono le loro preferenze su un piccolo insieme di tali variabili, valutate a coppie. La valutazione riguardante ogni tupla completa viene calcolata automaticamente sulla base delle coppie di valori che la compongono.
Quindi l'uso dei vincoli soft permette di aumentare le possibilità applicative dei metodi di aggregazione di preferenze di più agenti.

Questa tesi è organizzata come segue:

- Nel capitolo 1 sono introdotti i concetti principali riguardanti i vincoli "soft" e "hard" seguendo il formalismo algebrico descritto in [1, 2].

- Nel capitolo 2 viene presentato il metodo di aggregazione di preferenze introdotto da Conitzer e Sandholm in [6]. Si descrive inoltre il problema di generare un meccanismo che permetta di elaborare le valutazioni dei

singoli individui allo scopo di produrre un risultato che soddisfi l'intero gruppo di partecipanti.

- Nel capitolo 3 viene adattato il meccanismo di aggregazione di preferenze precedentemente descritto, allo scopo di risolvere i problemi vincolati multi-agente. Dato un insieme di variabili, si vuole produrre un meccanismo che aggreghi le valutazioni riportate da un gruppo di agenti sulle possibili assegnazioni di valori alle variabili.

- Nel capitolo 4 viene analizzato in maniera più dettagliata il problema affrontato e vengono descritte le scelte implementative.

- Nel capitolo 5 vengono descritti alcuni risultati sperimentali, e vengono tratte alcune considerazioni riguardanti l'efficienza nell'imporre i vincoli di non manipolabilità.

Capitolo 1

Problemi con vincoli

Nella rappresentazione classica, un problema con vincoli (*Constraint Satisfaction Problem -CSP-*) consiste in un insieme V di variabili, un insieme D di domini delle variabili (i valori ammessi per ciascuna variabile), e da un insieme C di vincoli sulle variabili. Ogni vincolo specifica quali assegnazioni siano permesse alle incognite coinvolte.
La soluzione di un problema con vincoli consiste in un assegnamento di valori a tutte le variabili, che rispetti i domini e soddisfi tutti i vincoli.

Consideriamo ad esempio il sistema:

$$\begin{cases} X + 2Y + Z = 5 \\ X + Y = Z \\ 3Y - Z = 7 \end{cases}$$

Lo si può vedere come un problema con vincoli in cui
$V = \{X, Y, Z\}$
$D = \{D_X, D_Y, D_Z\}$ con $D_X = D_Y = D_Z = \mathbb{Z}$ insieme dei numeri interi
$C = \{X + 2Y + Z = 5,\ X + Y = Z,\ 3Y - Z = 7\}$

Una soluzione è data da: $(X, Y, Z) = (-\frac{11}{9},\ \frac{26}{9},\ \frac{5}{3})$.

Questo tipo di rappresentazione, tuttavia, è soggetta a diverse limitazioni, dovute principalmente alla mancanza di flessibilità. Negli scenari della vita reale, infatti, sono vari i fattori che influiscono sulla risoluzione di un problema e che

non possono essere tenuti in considerazione attraverso una schematizzazione rigida, tipica dei CSP classici.
Per ovviare a queste difficoltà, il concetto di CSP è stato esteso mediante l'introduzione di un nuovo tipo di vincoli, detti *vincoli soft*.
In questo capitolo consideriamo una particolare classe di vincoli Soft, i vincoli *Fuzzy*, e la loro rappresentazione algebrica che permette di associare un grado di preferenza alle varie assegnazioni e di comporre i vincoli.

Introduciamo prima alcuni concetti basilari. (Per uno studio più approfondito si veda [1, 2])

1.1 CSP basati su semianelli

Il formalismo dei vincoli soft si basa sulla rappresentazione dei problemi CSP mediante una struttura matematica detta semianello (un dominio con due operazioni che soddisfano particolari proprietà).
Il dominio del semianello fornisce il livello di consistenza (che si può interpretare come un costo, un grado di preferenza, una probabilità o altro) e le due operazioni definiscono un modo per combinare tra loro i vincoli. La scelta di uno specifico semianello caratterizza ciascuna particolare istanza.

Definizione 1 *:(semianello)*
Un semianello è una tupla $< R, +, \times, 0, 1 >$ *formata da:*

- *un insieme* R *contenente gli elementi* 0 *e* 1*.*
- *un' operazione additiva* $+$*, commutativa, associativa e avente* 0 *come elemento neutro.*
- *un' operazione moltiplicativa* $\times$*, associativa, avente* 1 *come elemento neutro e tale che* $a \times 0 = 0 = 0 \times a \ \ \forall a \in R$ *(*0 *è detto elemento assorbente). Inoltre* $\times$ *è distributiva rispetto alla somma.*

Definizione 2 *:(c-semianello)*
Un c-semianello è una tupla $< R, +, \times, 0, 1 >$ *formata da:*

- *un insieme* R *contenente gli elementi* 0 *e* 1*.*

- *un' operazione additiva $+$, definita su sottoinsiemi di elementi di R come segue:*
  - *per ogni $a \in R$, $\sum(a) = a$;*
  - $\sum(0) = 0$ *e* $\sum(R) = 1$;
  - $\sum(\bigcup R_i,\ i \in S) = \sum(\sum(R_i),\ i \in S)$ *per ogni insieme di indici S;*
- *un' operazione moltiplicativa $\times$, binaria, associativa e commutativa, avente 1 come unità e 0 come elemento assorbente; questa operazione moltiplicativa è distributiva rispetto all'operazione additiva $+$.*

Poichè l'operazione $+$ è definita su ogni sottoinsieme di elementi di R e non su coppie o n-uple tale operazione è commutativa, associativa ed idempotente; 0 è il suo elemento neutro e 1 l'elemento assorbente. Dunque ogni c-semianello è un particolare semianello. L'idempotenza dell'operazione $+$ permette di definire un ordinamento parziale[1] sull'insieme R, utile a confrontare i diversi elementi: $a \leq b \Leftrightarrow a + b = b$.
Intuitivamente: $a \leq b$ significa che b è migliore di a, e quindi tra a e b prevale b.
Il fatto che 0 sia l'elemento neutro dell'operazione additiva implica che 0 è il minimo elemento rispetto all'ordine appena definito.
Infatti $a + 0 = a\ \forall a \in R$ implica che $0 \leq a\ \forall a \in R$.
Si può inoltre osservare che le $+$ e $\times$ sono monotone rispetto all'ordine parziale $\leq$, ossia $a \leq a'$ implica $a + b \leq a' + b$ e $a \times b \leq a' \times b$.

Definizione 3 *:(sistema vincolato)*
Un sistema vincolato $CS =< S, D, V >$, è una terna formata da un c-semianello S, un insieme finito di valori D ed un insieme ordinato V di variabili.

Un vincolo connette n variabili, con $n \leq \mid V \mid$, ed assegna ad ogni n-upla di valori di D un corrispondente elemento di R, che può essere interpretato come peso, costo o altro.
La struttura di un vincolo si può vedere in Fig. 1.1.

[1] Un ordine parziale è una relazione binaria, riflessiva: $a \leq a\ \forall\ a \in R$, transitiva: $a \leq b,\ b \leq c \Rightarrow a \leq c\ \forall\ a, b, c \in R$ e antisimmetrica: $\forall\ a, b \in R\ \ a \leq b,\ b \leq a \Rightarrow a = b$.

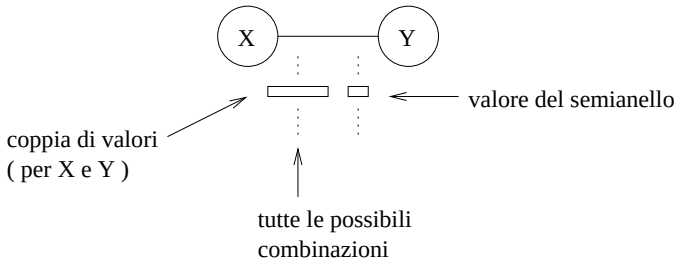


Figura 1.1: Esempio di vincolo binario basato su un semianello

Definizione 4 *:(vincolo)*
Dato un sistema vincolato $CS =< S, D, V >$*, dove* $S =< R, +, \times, 0, 1 >$*, un vincolo su* CS *è una coppia* $< def, con >$*, dove*

- $con \subseteq V$*, è detto il tipo del vincolo;*
- $def : D^k \longrightarrow R$ *(dove* k *è la cardinalità di con) è detta il valore del vincolo.*

Definizione 5 *:(problema con vincoli)*
Dato un sistema vincolato $CS =< S, D, V >$*, un problema con vincoli* P *su* CS *è una coppia* $P =< C, con >$*, dove* C *è un insieme di vincoli su* CS *e* $con \subseteq V$*.*

Definizione 6 *:(proiezione di una k-upla)*
Dato un sistema vincolato $CS =< S, D, V >$ *con* V *totalmente ordinato mediante l'ordine* $\prec$[2]*, consideriamo ogni k-upla di valori in* D*:* $t =< t_1, \ldots, t_k >$ *e due insiemi* $W = \{w_1, \ldots, w_k\}$ *e* $W' = \{w'_1, \ldots, w'_m\}$ *tali che* $W' \subseteq W \subseteq V$ *e* $w_i \prec w_j$ *se* $i \leq j$ *e* $w'_i \prec w'_j$ *se* $i \leq j$*.*
Allora la proiezione di t *da* W *a* W' *:* $t \downarrow^W_{W'}$ *è definita da* $t' =< t'_1, \ldots, t'_m >$ *con* $t'_i = t_j$ *se* $w'_i = w_j$*.*

Definizione 7 *:(combinazione)*
Dato un sistema vincolato $CS =< S, D, V >$*, dove* $S =< R, +, \times, 0, 1 >$*, e due vincoli* $c_1 =< def_1, con_1 >$ *e* $c_2 =< def_2, con_2 >$ *su* CS*, la combinazione*

[2]L'ordinamento totale $\prec$ è una relazione binaria che si ricava da $\leq$ definendo: $a \prec b \Leftarrow a \leq b$ *e* $a \neq b$. Valgono le proprietà di $\leq$ (riflessiva, transitiva e antisimmetrica) ed è totale: $\forall a, b \in R,\ a \neq b,\ a \prec b$ oppure $b \prec a$

di c_1 *e* c_2 *:* $c_1 \otimes c_2$ *è il vincolo* $c =< def, con >$ *tale che:*

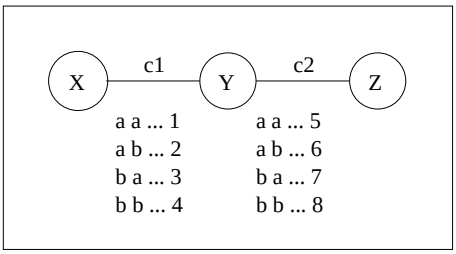
$$\begin{cases} con = con_1 \cup con_2 \\ def(t) = def_1(t \downarrow_{con_1}^{con}) \times def_2(t \downarrow_{con_2}^{con}) \end{cases}$$

Poichè $\times$ è sia commutativa che associativa, anche $\otimes$ lo è. Questa operazione può quindi essere facilmente estesa a più di due elementi.

Definizione 8 *:(proiezione)*
Dato un sistema vincolato $CS =< S, D, V >$*, dove* $S =< R, +, \times, 0, 1 >$*,un vincolo* $c =< def, con >$ *su* CS *e un insieme* $I \subset V$ *di variabili, la proiezione di* c *su* I*:* $c \Downarrow_I$*, è il vincolo* $< def', con' >$ *su* CS *tale che:*

$$\begin{cases} con' = I \cap con \\ def'(t') = \sum_{\{t | t \downarrow_{I \cap con}^{con} = t'\}} def(t) \end{cases}$$

Esempio: La seguente figura mostra l'effetto di queste due operazioni su un problema con due vincoli c_1 e c_2



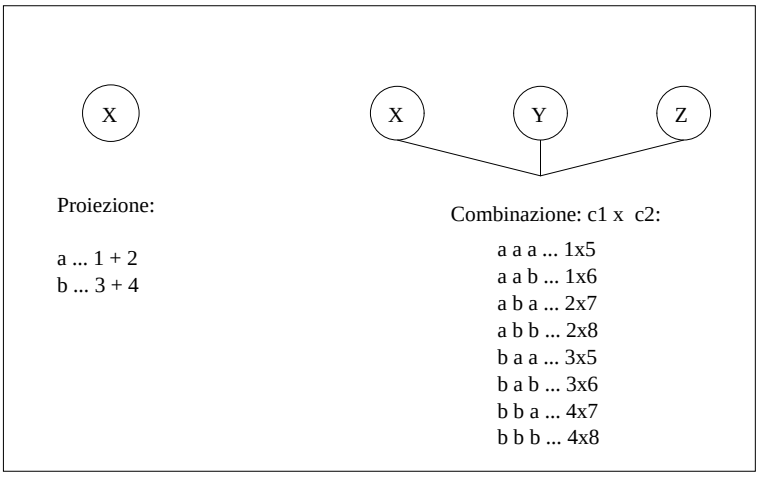


Figura 1.2: Effetti delle operazioni di proiezione e combinazione su un problema con due vincoli e tre variabili aventi dominio {a,b}

1.1.1 CSP classici

Un problema CSP classico è dato da un insieme di variabili e di vincoli in cui ogni vincolo specifica i valori permessi per le variabili.
Per rappresentarlo mediante la struttura algebrica appena descritta, consideriamo il semianello:

$$< \{true, false\}, \vee, \wedge, false, true >$$

$R = \{true, false\}$ è costituito da due soli elementi data la rigidità di questo tipo di problemi: ogni assegnazione di valori alle variabili, infatti, o è permessa (vi corrisponde il valore true), oppure è negata (vi corrisponde il valore false). Si fissa come operazione moltiplicativa l'*and* logico[3]: $\wedge$.
Infine si costruisce la funzione di proiezione assumendo come operazione additiva l'*or* logico[4]: $\vee$.

Esempio: Supponiamo che le variabili del problema siano X,Y,Z, aventi ciascuna come dominio l'insieme:$D = \{a, b, c\}$. Queste variabili siano tra loro vincolate in modo che le coppie permesse da ciascun vincolo binario siano date

[3] $true \wedge true = true$, $true \wedge false = false$,
$false \wedge true = false$, $false \wedge false = false$

[4] $true \vee true = true$, $true \vee false = true$,
$false \vee true = true$, $false \vee false = false$

dalla figura 1.3:

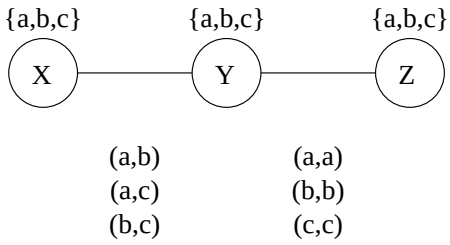


Figura 1.3: Un problema CSP

Assegnamo un valore ad ogni terna di possibili assegnazioni nel seguente modo: presa in considerazione ogni coppia di valori corrispondente ad uno dei due vincoli(es: (X,Y)=(a,b)), le assegnamo valore 1 se tale coppia è permessa dal vincolo, 0 altrimenti:

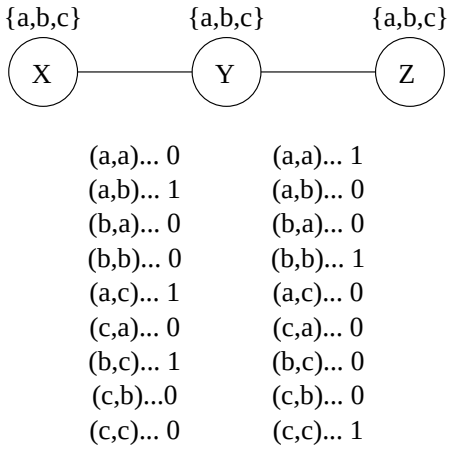


Figura 1.4: Valori dell'anello

Le triple risultanti avranno dunque un valore dato dalla combinazione dei valori di ciascuna coppia.
Considerando che:

$$1 \wedge 1 = 1$$
$$1 \wedge 0 = 0$$
$$0 \wedge 1 = 0$$
$$0 \wedge 0 = 0$$

si ottiene che le terne aventi valore 1 sono: (a,b,b), (a,c,c) (b,c,c) che si vede facilmente essere le uniche soddisfacenti tutti i vincoli.

1.1.2 Fuzzy CSP

Consideriamo ora un caso particolare di problemi con vincoli: i CSP Fuzzy, o FCSP. Abbiamo visto nell'introduzione come i vincoli Fuzzy estendano il concetto classico di CSP permettendo l'assegnazione di preferenze. Formalizziamo ora quanto detto sopra.
I valori di preferenza appartengono all'intervallo [0,1], dove 1 rappresenta il livello più alto di gradimento (cioè l'assegnazione è auspicabile) e 0 rappresenta il livello più basso (l'assegnazione non è permessa).
Valori intermedi rappresentano un ordine di importanza dei vincoli.
La soluzione ottima di un FCSP consiste nell'insieme delle n-uple di valori di tutte le variabili (un'assegnazione a tutte le variabili) aventi preferenza massima. Il livello assegnato ad una n-upla (tramite l'operatore $\times$) è la più piccola (la minima) tra le preferenze delle assegnazioni che la compongono.
La struttura a semianello da considerare quando si parla di FCSP è quindi:

$$< \{x \mid x \in [0,1]\}, max, min, 0, 1 >$$

L'ordinamento dei valori dell'anello coincide con l'ordine $\leq$ nei reali.

Esempio:
Siano X,Y,Z tre variabili aventi i seguenti domini:
$D_X = \{1, 2\}$
$D_Y = \{3, 4\}$
$D_Z = \{1, 2, 3, 4\}$
Sia $C = \{X \leq Z, Y \leq Z\}$ l'insieme dei vincoli sulle variabili, tale che:

(X,Z)	$Preferenza$	(Y,Z)	$Preferenza$
$(1,1)$	1	$(3,1)$	0.3
$(1,2)$	1	$(3,2)$	0.6
$(1,3)$	1	$(3,3)$	1
$(1,4)$	1	$(3,4)$	1
$(2,1)$	0.4	$(4,1)$	0.25
$(2,2)$	1	$(4,2)$	0.5
$(2,3)$	1	$(4,3)$	0.75
$(2,4)$	1	$(4,4)$	1

Tutte le soluzioni possibili, con le rispettive preferenze, sono date dalle terne (X,Y,Z):

$(1,3,1) \ldots min(1,0.3) = 0.3$ $(2,3,1) \ldots min(0.4,0.3) = 0.3$
$(1,3,2) \ldots min(1,0.6) = 0.6$ $(2,3,2) \ldots min(1,0.6) = 0.6$
$(1,3,3) \ldots min(1,1) = 1$ $(2,3,3) \ldots min(1,1) = 1$
$(1,3,4) \ldots min(1,1) = 1$ $(2,3,4) \ldots min(1,1) = 1$
$(1,4,1) \ldots min(1,0.25) = 0.25$ $(2,4,1) \ldots min(0.4,0.25) = 0.25$
$(1,4,2) \ldots min(1,0.5) = 0.5$ $(2,4,2) \ldots min(1,0.5) = 0.5$
$(1,4,3) \ldots min(1,0.75) = 0.75$ $(2,4,3) \ldots min(1,0.75) = 0.75$
$(1,4,4) \ldots min(1,1)) = 1$ $(2,4,4) \ldots min(1,1) = 1$

L'insieme delle soluzioni ottime è dato da:

$(1,3,3) \ldots 1$
$(1,3,4) \ldots 1$
$(1,4,4) \ldots 1$
$(2,3,3) \ldots 1$
$(2,3,4) \ldots 1$
$(2,4,4) \ldots 1$

Capitolo 2

Automated Mechanism Design

2.1 Progettazione di un meccanismo

In molti problemi di intelligenza artificiale (e non solo), un risultato comune deve essere scelto a partire dalle preferenze riportate da un gruppo di agenti. Generalmente, però, ci si trova a dover aggregare tra loro preferenze contrastanti allo scopo di ottenere un risultato socialmente desiderabile. Può essere, ad esempio, un presidente, un piano comune, l'allocazione di alcuni beni o risorse,....

Questo tipo di problema si presenta in tutti i casi in cui si hanno agenti interessati al proprio vantaggio: siano essi persone, aziende o altro. I meccanismi di aggregazione delle preferenze che possono essere prodotti spaziano tra protocolli di voto, aste, commercio elettronico, regolamentazioni di divorzio e sistemi di valutazione collettivi, per citarne alcuni.

Purtroppo la maggior parte di essi soffre di manipolabilità: poichè ogni agente riferisce le proprie valutazioni secondo il personale interesse, può succedere che il risultato selezionato dal meccanismo non rappresenti veramente una soluzione ottima, ad esempio se un agente non riporta le valutazioni in modo sincero. Essendo questo tipo di meccanismi progettati per produrre un risultato utile all'intero gruppo di partecipanti, il fenomeno della manipolazione rappresenta un grave problema perchè porta facilmente alla selezione di un risultato che non coincide con il bene comune.

In particolare, secondo un teorema dimostrato da Gibbard e da Satterwaite (si

veda [7, 8]), sotto ogni schema non dittatoriale[1] di aggregazione di preferenze, con almeno 3 risultati possibili, ci sono dei valori di utilità (per problemi di selezione con almeno 3 risultati possibili) per i quali una strategia ottima per un agente consiste nel mentire.
Il "Mechanism design" consiste nel progettare un meccanismo (ossia definire le regole) che motivi gli agenti a riportare in modo sincero le proprie preferenze e che produca un risultato desiderabile, secondo un dato obiettivo. Esempi di possibili obiettivi sono il benessere sociale,(somma dell'utilità di ciascun agente), il guadagno di un venditore, l'imparzialità o una combinazione di questi.

2.1.1 Progettazione manuale

Normalmente la progettazione di un meccanismo viene svolta manualmente. Il progettista usa l'esperienza e l'intuito per generare un insieme di regole che possano portare al risultato richiesto, e poi implementa un meccanismo specifico per la particolare istanza in questione.
In questo modo sono stati progettati alcuni meccanismi canonici, ciascuno rivolto ad una particolare classe di problemi e per un obiettivo specifico.
Proprio il fatto di essere stati progettati per risolvere problemi specifici rende questi meccanismi inapplicabili nella soluzione di istanze diverse da quelle previste.
Alcuni, come il meccanismo VCG di Vickrey-Clarke-Groves [11, 13, 12], massimizzano solo il benessere sociale; non sono quindi applicabili a quei problemi in cui il progettista vuole massimizzare anche il proprio interesse personale (com'è richiesto, ad esempio, nel commercio elettronico).
Al contrario, i meccanismi che focalizzano la propria attenzione sull'interesse del progettista sono applicabili solo in situazioni molto ristrette: l'asta di Myerson [16], ad esempio, massimizza il guadagno atteso ma si può usare solo nella vendita di un unico oggetto. L'asta di Maskin e Riley [17], invece, permette la vendita di varie unità ma di un unico articolo.
Permettono la massimizzazione dei pagamenti, ma nella pratica ci possono essere gestori d'asta interessati ad assegnare l'articolo ad un particolare offerente.

[1] Uno schema di aggregazione di preferenze è detto dittatoriale quando uno degli agenti determina il risultato, indipendentemente da quanto dichiarato dagli altri.

Altra perdita di generalità si ha per il fatto che questi meccanismi suppongono che le preferenze degli agenti siano quasi lineari. Questo significa che per ogni agente A_i si può scrivere una funzione di utilità u_i legata unicamente al risultato finale O ed alla somma da pagare π_i: $u_i(O, \pi_i)$,mentre si suppone, in modo molto restrittivo, che la valutazione v_i del risultato non dipenda dal pagamento, che ogni agente non si preoccupi del pagamento effettuato dagli altri, e che tutti i partecipanti siano neutrali al rischio.
Infine, per varie classi di problemi, alcuni risultati di impossibilità hanno dimostrato come non vi possano esistere meccanismi soddisfacenti.

2.1.2 Progettazione automatizzata

In forte contrasto con la progettazione manuale del meccanismo, Conitzer e Sandholm [9] hanno introdotto un metodo sistematico, denominato *"Automated Mechanism Design"*, in cui un meccanismo viene generato automaticamente ad ogni istanza specifica.
Oltre a spostare la difficoltà di progettazione dall'uomo alla macchina, la progettazione automatizzata può essere riapplicata nelle classi di problemi finora studiati con la progettazione manuale, migliorando in tal modo i meccanismi già noti (in termini di risultato o di maggiore garanzia contro la manipolabilità). Poichè si basa sulle particolari informazioni riguardanti gli agenti, quersto metodo permette di aggirare i risultati di impossibilità o di minimizzarne la penalità.

2.2 Il problema computazionale

La progettazione del meccanismo è modellata come un problema di ottimizzazione: [9, 10] si considera un gruppo di N agenti, con le rispettive valutazioni e si produce un meccanismo per aggregare le loro preferenze che massimizzi una particolare funzaione obiettivo.

Definizione 9 *:*
Nella progettazione automatica di un meccanismo sono dati:

- *un insieme finito di risultati O;*
- *un insieme finito di agenti $\{1, 2, \ldots, N\}$;*

- *per ogni agente A_i:*
  - *un insieme finito di tipi T_i. I tipi di un agente corrispondono alla valutazione che un agente esprime per un particolare risultato;*
  - *una distribuzione di probabilità P_i su T_i. Essa è dovuta al fatto che al momento della progettazione non sono note le dichiarazioni dei tipi che gli agenti faranno nell'istanza specifica.*
  - *una funzione di utilità u_i: $T_i \times O \longrightarrow \mathbb{R}$, che assegna un valore numerico ad ogni evento, legato alla valutazione che era stata fatta su di esso.*
- *una funzione obiettivo;*
- *una specifica di quali strumenti si possano utilizzare: se sono permessi i pagamenti e la randomizzazione.*

Ci sono diverse funzioni obiettivo che si possono considerare, ad esempio il benessere sociale (in cui il progettista cerca di massimizzare le utilità degli agenti), o l'utilità minima (in cui si cerca di massimizzare la minima utilità ottenuta da ciascun agente).
In entrambi questi casi il progettista è *benevolo* perchè ricerca, in un certo qual senso, la felicità collettiva degli agenti.
In altri casi un progettista può essere *self-interested*: si preoccupa soltanto del risultato ottenuto e dei pagamenti riscossi.
Formalmente, un progettista self-interested ha una funzione obiettivo:
$g(o) + \sum_{i=1}^{N} \pi_i$, dove $g : O \longrightarrow \mathbb{R}$ indica la preferenza del progettista sui risultati e π_i è il pagamento effettuato dall'agente A_i. Nel caso in cui valga costantemente $g = 0$ si massimizzano i pagamenti.

Un meccanismo può essere deterministico o randomizzato, a seconda che in fase di esecuzione il risultato prodotto sia unico, oppure una perticolare distribuzione di probabilità su tutti i risultati possibili:

Definizione 10 Meccanismo deterministico:
Un meccanismo deterministico senza pagamenti consiste in una funzione
$o : T_1 \times T_2 \times \ldots \times T_N \longrightarrow O$

che, una volta note le preferenze $(t_1 \dots t_N)$ *degli agenti, ritorna il risultato* $o(t_1 \dots t_N)$.

Definizione 11 Meccanismo randomizzato:
Un meccanismo randomizzato senza pagamenti consiste in una funzione
$p : T_1 \times T_2 \times \dots \times T_N \longrightarrow P(O)$
che assegna ad ogni vettore di tipi $(t_1 \dots t_N)$ *una distribuzione di probabilità sull'insieme dei possibili risultati, con* p_i = *probabilità del risultato* o_i: $p_i \in [0,1]$ *e* $p_1 + p_2 + \dots + p_k = 1$

In entrambi i casi, se sono permessi pagamenti, viene definita per ogni agente A_i un'ulteriore funzione:
π_i: $T_1 \times T_2 \times \dots \times T_N \longrightarrow \mathbb{R}$
che rappresenta il pagamento che l'agente deve sostenere in funzione dei tipi dichiarati dall'intero gruppo di partecipanti al sistema.

Ad esempio: siano A_1,A_2 due agenti, t_i il tipo dell'agente A_i e O_1,O_2,...,O_k i possibili risultati. Un meccanismo deterministico può essere schematizzato con una tabella a due entrate (i tipi dei due agenti) in cui ogni cella contiene il risultato prodotto dalla selezione dei tipi dichiarati da ciascun agente, come mostrato in Tab. 2.1, un meccanismo randomizzato invece rirorna ogni possibi-

	...	$A_1 t_i$	...	
⋮		⋮		
$A_2 t_j$	...	$o(t_i, t_j)$	...	
⋮		⋮		

Tabella 2.1: Il meccanismo deterministico che viene generato per un problema con due agenti: un determinato risultato in funzione dei tipi.

le risultato e la probabilità di questo di verificarsi, come mostrato in Tab. 2.2: $[P_1 : o_1]$,...,$[P_k : o_k]$.

Ci sono due tipi di vincoli che il progettista deve considerare nella costruzione del meccanismo al fine di renderlo non manipolabile: i vincoli di razionalità individuale (*Individual Rationality -IR-*) ed i vincoli di compatibilità (*Incenti-*

	...	A_1t_i	...	
$\vdots$		$\vdots$		
A_2t_j	...	$[P_1 : O_1]$,...,$[P_k : O_k]$		
$\vdots$		$\vdots$		

Tabella 2.2: Il meccanismo randomizzato che viene generato per un problema con due agenti: una distribuzione di probabilità su tutti i possibili risultati.

ve compatibilità -IC-).

2.2.1 Vincolo di partecipazione: IR

Il primo tipo di vincoli impone che l' utilità di ogni agente nel partecipare al problema sia maggiore od uguale all'utilità che l'agente riceverebbe se non vi partecipasse (altrimenti non vi parteciperebbe). Questa classe di vincoli è chiamata Individual Rationality". Più in dettaglio, il vincolo IR può essere:

- "*ex ante IR*": non conoscendo niente, neppure il proprio tipo, è conveniente per un agente partecipare.
- "*ex interim IR*": conoscendo solo il proprio tipo, è conveniente per un agente partecipare.
- "*ex post IR*": conoscendo già quanto dichiarato dagli altri agenti, è conveniente per un agente partecipare.

In questo nostro lavoro non considereremo il caso "*ex ante IR*" in quanto, presupponendo che nessun agente conosca il proprio tipo, non avrebbe senso ricercare un metodo di aggregazione delle preferenze, non esistendo le preferenze stesse.
Tralasciando allora la prima formulazione del vincolo IR, vediamo in dettaglio le altre due.

Supponiamo, senza perdita di generalità, che l'utilità di un agente nel non prendere parte al meccanismo sia nulla. Allora:

Definizione 12 (ex interim IR)
Un meccanismo deterministico *è "ex interim IR" se, per ogni agente A_i e per ogni tipo $t_i \in T_i$ si ha:*

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[u_i(t_i, o(t_1,\ldots,t_N) - \pi_i(t_1,\ldots,t_N)\Big] \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P_j(t_j) \geq 0$$

dove $P(t_j)$ è la probabilità che il tipo dichiarato dall'agente A_j sia t_j.

Un meccanismo randomizzato *è "ex interim IR" se, per ogni agente A_i e per ogni tipo $t_i \in T_i$:*

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\sum_{o|(t_1,\ldots,t_N)} \big[u_i(t_i,o) - \pi_i(t_1,\ldots,t_N)\big] \times P(o)\Big] \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P_j(t_j) \geq 0$$

I termini riguardanti i pagamenti sono nulli ove questi non sono permessi.

Torniamo ancora, per chiarire, all'esempio precedentemente citato. I due agenti non conoscono quanto dichiarato dalla controparte e quindi hanno convenienza a partecipare solo se ogni loro possibile scelta comporta un'utilità attesa[2], al variare di tutti i tipi dell'avversario, non negativa (trascuriamo i pagamenti). Allora, per l'agente A_1 la somma pesata delle utilità $u_1(t_i, o)$ al variare dei risultati o nella colonna corrispondente a t_i deve essere non negativa per ogni tipo $t_i \in T_1$.
Analogamente, per l'agente A_2 deve essere non negativa l'utilità attesa lungo ogni riga.
Se il meccanismo è randomizzato bisogna tener conto del fatto che in ogni cella della tabella compaiono più risultati, con le rispettive probabilità.

Definizione 13 (ex post IR)
Un meccanismo deterministico *è "ex post IR" se, per ogni agente A_i e per ogni*

[2]L'utilità attesa è il valore medio probabilistico dell'utilità di ciascun agente, ossia la somma dell'utilità di ciascun risultato, pesata con la probabilità di questo di verificarsi. Se X è una variabile che può assumere i valori: 5 con probabilità 0.3, 10 con probabilità 0.4 e 15 con probabilità 0.3, il valore atteso di X vale $E(X) = 5 \times P(X = 5) + 10 \times P(X = 10) + 15 \times P(X = 15) = 1.5 + 4 + 4.5 = 10$

vettore di tipi $(t_1 \dots t_N) \in T_1 \times T_2 \times \dots \times T_N$ *si ha:*

$$u_i(t_i, o(t_1 \dots t_N)) - \pi_i(t_1, \dots, t_N) \geq 0$$

Un meccanismo randomizzato *è "ex post IR" se, per ogni agente* A_i *e per ogni vettore di tipi* $(t_1 \dots t_N) \in T_1 \times T_2 \times \dots \times T_N$ *si ha:*

$$\sum_{o|t_1 \dots t_N} \Big[u_i(t_i, o) - \pi_i(t_1, \dots, t_N)\Big] \times P(o) \geq 0$$

I termini riguardanti i pagamenti sono nulli ove questi non sono permessi.

Imponendo il vincolo "ex post IR" si presuppone che ogni agente conosca il tipo dichiarato dalla controparte, quindi sarà motivato a partecipare solo se l'utilità di ogni risultato presente nella tabella è non negativa. Per un meccanismo randomizzato bisogna verificare che l'utilità attesa in ogni cella della tabella prodotta sia non negativa per entrambi gli agenti.

2.2.2 Concetto di soluzione: IC

Il secondo tipo di vincolo dichiara che gli agenti non dovrebbero mai essere motivati a falsificare il proprio tipo.
Le due varianti più comuni (dette concetti di soluzione) sono l'implementazione nelle strategie dominanti (in cui ogni agente conosce il tipo riportato dagli altri e la sua strategia ottima è quella veritiera), e l'implementazione negli equilibri di Bayes-Nash [4, 5] (in cui non sono noti a priori i tipi riportati dagli altri):

Definizione 14 *Si dice che un meccanismo implementa le funzioni di risultato e di pagamento nelle strategie dominanti, se il metodo veritiero è sempre ottimale, anche quando sono noti i tipi riportati dagli altri agenti.*
Formalmente, per ogni agente A_i *e per ogni vettore di tipi* $(t_1 \dots t_i \dots t_N) \in T_1 \times \dots \times T_i \times \dots \times T_N$ *e ogni tipo alternativo* $\hat{t}_i$ *dell'agente* A_i *si ha:*

$$\begin{aligned} &u_i(t_i, o(t_1 \dots t_i \dots t_N)) - \pi_i(t_1 \dots t_i \dots t_N) \quad \geq \\ &u_i(t_i, o(t_1 \dots \hat{t}_i \dots t_N)) - \pi_i(t_1 \dots \hat{t}_i \dots t_N) \end{aligned}$$

Nel caso di meccanismo randomizzato si ha:

$$\sum_{o|t_1 \ldots t_i \ldots t_N} \Big[u_i(t_i, o(t_1 \ldots t_i \ldots t_N)) - \pi_i(t_1 \ldots t_i \ldots t_N)\Big] \times P(o) \quad \geq$$
$$\sum_{o|t_1 \ldots \hat{t}_i \ldots t_N} \Big[u_i(t_i, o(t_1 \ldots \hat{t}_i \ldots t_N)) - \pi_i(t_1 \ldots \hat{t}_i \ldots t_N)\Big] \times P(o)$$

Ritornando al nostro esempio con 2 agenti, sia $\bar{t}$ il tipo dichiarato (in modo veritiero) dal secondo agente e sia t_i i tipo veritiero di A_1. L'utilità del risultato che si ottiene da t_i e $\bar{t}$ deve essere maggiore o uguale dell'utilità di ogni risultato lungo la stessa riga (corrispondente a $\bar{t}$), valutata rispetto al tipo t_i: $u_1(t_i, o(t_i, \bar{t}) \geq u_1(t_i, o(t_j, \bar{t}) \ \forall j \neq i$. Infatti, se esistesse un tipo t_j tale che $u_1(t_i, o(t_j, \bar{t}) \geq u_1(t_i, o(t_i, \bar{t})$, l'agente A_1 srebbe incentivato a riportare falsamente t_j, aumentando così la propria utilità.

Quindi nelle strategie dominanti è ottimale riferire il vero, indifferentemente da quanto dichiarato dagli altri agenti. Se, invece, dichiarare il vero è ottimale solo quando non si conoscono i tipi riportati dalla controparte, allora si ha l'implementazione negli equilibri di Bayes-Nash.

Definizione 15 *Si dice che un meccanismo implementa le funzioni di risultato e di pagamento negli equilibri di Bayes-Nash, se il metodo veritiero è sempre ottimale per un agente che non conosce nulla riguardo ai tipi riportati dagli altri agenti, supposto che essi non mentano.*
Formalmente, per ogni agente A_i, ogni tipo $t_i \in T_i$ e ogni tipo alternativo $\hat{t}_i \in T_i \setminus \{t_i\}$ si ha:

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[u_i(t_i, o(t_1 \ldots t_i \ldots t_N)) - \pi_i(t_1 \ldots t_i \ldots t_N)\Big] \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P_j(t_j) \quad \geq$$
$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots t_N)|t_i} \Big[u_i(t_i, o(t_1 \ldots \hat{t}_i \ldots t_N)) - \pi_i(t_1 \ldots \hat{t}_i \ldots t_N)\Big] \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P_j(t_j)$$

Nel caso di meccanismo randomizzato si ha:

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\sum_{o|t_1\ldots t_i\ldots t_N} u_i(t_i,o) - \pi_i(t_1\ldots t_i\ldots t_N) \times P(o) \Big] \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P_j(t_j) \quad \geq$$

$$\sum_{(t_1\ldots t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\sum_{o|t_1\ldots \hat{t}_i\ldots t_N} u_i(t_i,o) - \pi_i(t_1\ldots \hat{t}_i\ldots t_N) \times P(o) \Big] \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P_j(t_j)$$

Supponiamo che il tipo veritiero dell'agente A_1 sia t_i. Allora l'utilità attesa lungo ogni altra colonna valutata rispetto al tipo t_i è $\sum_{tipo\ di\ A_2} u_1(t_i, o(t_j, tipo\ di\ A_2) \times P(tipo\ di\ A_2)$, e per $j \neq i$ deve essere minore dell'utilità attesa della colonna con t_j, altrimenti A_1 sarebbe motivato a riportare un tipo falso, allo scopo di aumentare la propria utilità. La stessa condizione vale per il meccanismo randomizzato, in cui però bisogna tener conto delle probabilità dei vari risultati all'interno di ogni cella. In modo simmetrico per il secondo agente: l'utilità attesa lungo la riga corrispondente al tipo veritiero deve essere maggiore o uguale all'utilità attesa lungo ogni altra riga, valutata rispetto quello stesso tipo.

2.2.3 Il problema di ottimizzazione

Introdotti questi concetti principali, vediamo come sia possibile generare il meccanismo richiesto: una volta assegnato il problema computazionale, fissato un vincolo di partecipazione e un concetto di soluzione, si tratta di risolvere un problema di ottimizzazione.

Definizione 16 :(AUTOMATED MECHANISM DESIGN - AMD -)
Siano dati:

- *un problema computazionale di progettazione di meccanismo,*
- *una nozione di vincolo IR (ex interim, ex post o nessuna),*
- *un concetto di soluzione (strategie dominanti o equilibri di Bayes-Nash),*
- *un insieme di strumenti disponibili (pagamenti, randomizzazione,...),*
- *un valore obiettivo G.*

Si chiede: esiste un meccanismo del tipo specificato, che soddisfa i vincoli imposti (IR e IC) ed assegna un valore atteso maggiore od uguale a G alla funzione obiettivo?

2.3 Complessità

Riassiumiamo brevemente alcune considerazioni riguardanti la complessità di progettare automaticamente un meccanismo. Per un'analisi più approfondita si vedano gli articoli [19] e [20].
La progettazione di un meccanismo deterministico senza pagamenti, che massimizzi una funzione obiettivo è un problema $\mathcal{NP}$-completo[3], sia nel caso in cui il progettista sia benevolo, che nel caso in cui si abbia un progettista interessato al proprio vantaggio. Se sono permessi i pagamenti la complessità diminuisce se la funzione obiettivo è il benessere sociale, mentre rimane $\mathcal{NP}$-completa quando si massimizzano i pagamenti riscossi.
Il problema di generare un meccanismo randomizzato, invece, può essere risolto in tempo polinomiale (per qualsiasi numero di agenti) usando la programmazione lineare (LP). Per ogni numero costante di agenti, infatti, il numero di variabili decisionali (sono le incognite del problema: le variabili decisionali nei nostri modelli LP saranno le probabilità deirisultati di essere scelti, a partire da un carto vettore di tipi) è polinomiale nel numero di tipi e nel numero di risultati, così come il numero di vincoli (IC e IR). Ne deriva che il problema di generare un meccanismo randomizzato sta in $\mathcal{P}$[4].

2.4 Esempio: Regolamentazioni di divorzio

Una semplice applicazione di AMD è data dalle regolamentazioni di divorzio [18].
Vediamo diverse varianti del problema di progettare un meccanismo e le rispettive soluzioni ottimali generate tramite l'esecuzione di AMD.

[3] È un problema risolvibile in tempo polinomiale (in funzione della grandezza dell'input) con un algoritmo non deterministico.

[4] È risolvibile in tempo polinomiale con un algoritmo deterministico.

In un primo caso esaminiamo l'esempio in cui vi è un arbitro benevolo, unicamente interessato a massimizzare il benessere sociale; poi consideriamo un arbitro benevolo che utilizza i pagamenti per incentivare gli agenti a riportare il vero; infine il caso in cui vi è un arbitro interessato a massimizzare la somma dei pagamenti raccolti, pur motivando gli agenti a partecipare.

2.4.1 Un arbitro benevolo

Una coppia M,F sta ottenendo il divorzio. Essi possiedono in comune un quadro ed un giudice deve decidere a chi assegnarlo.
Ci sono 4 possibili scelte:
(M) Il marito ottiene il quadro.
(F) La moglie ottiene il quadro.
(E) Il quadro rimane di proprietà di entrambi e viene conservato in un museo.
(B) Il quadro viene bruciato.

Entrambi i coniugi hanno la stessa distribuzione di probabilità sugli stessi due tipi:

$$\begin{cases} 0.8 & \text{L} \quad \text{(indica scarso interesse per l'oggetto in questione)} \\ 0.2 & \text{H} \quad \text{(indica un forte interesse per l'oggetto in questione)} \end{cases}$$

Per massimizzare il benessere sociale, l'arbitro vorrebbe dare il quadro a chi dei due è più interessato ad averlo. L'accertamento delle preferenze però non è così banale in quanto entrambi preferiscono avere il quadro al non averlo, anche quando non ne sono molto interessati.

La funzione di utilità è (per entrambi):

u(L, ricevo il quadro)	$= 2$	u(H, ricevo il quadro)	$= 100$
u(L, lo riceve il partner)	$= 0$	u(H, lo riceve il partner)	$= 0$
u(L, museo)	$= 1$	u(H, museo)	$= 50$
u(L, bruciato)	$= -10$	u(H, bruciato)	$= -10$

Si assuma, per ora, che i pagamenti non siano permessi, non vi sia randomizzazione e sia richiesta un'implementazione nelle strategie dominanti.
Il meccanismo prodotto si può schematizzare con la tabella 2.3.

	M L	M H
F L	O_1	O_2
F H	O_3	O_4

Tabella 2.3: Il meccanismo prodotto: ML, MH, FL, FH sono i tipi del primo e secondo agente, mentre O_i è il risultato prodotto.

Poichè è richiesta un implementazione nelle strategie dominanti, si suppone che ogni agente conosca il tipo riportato dalla parte opposta. Per motivare entrambi a partecipare bisogna allora imporre il vincolo "ex post IR":
Si deve avere:

- u_M(tipo di M,o(tipo di M, tipo di F))≥ 0
- u_F(tipo di F,o(tipo di M, tipo di F))≥ 0

per ogni tipo di M e di F.
Quello che si deve imporre, allora, è che, in ogni cella della tabella, l'utilità di ciascun risultato in funzione dei tipi che permettono di ottenerlo sia non negativa. Poichè l'unico risultato che può fornire un'utilità negativa è B (bruciare il quadro), non sono permesse tutte le tabelle in cui compare tale soluzione. Quindi $O_1 \neq B, O_2 \neq B, O_3 \neq B, O_4 \neq B$.
Ogni agente, poi, deve essere motivato a dire il vero, pur conoscendo già quanto dichiarato dal compagno. Supponiamo che entrambi abbiano come tipo veritiero L. Si deve avere $u_M(\mathrm{L}, O_1) \geq u_M(\mathrm{L}, O_2)$ altrimenti M sarebbe motivato a riportare falsamente H. Allo stesso modo $u_F(\mathrm{L}, O_1) \geq u_M(\mathrm{L}, O_3)$, altrimenti F riporterebbe il tipo H.
Ad esempio, un meccanismo della forma rappresentata in Tab. 2.4:

	M L	M H
F L	E	M
F H	F	E

Tabella 2.4: Meccanismo non accettabile per questa istanza del problema.

non può essere accettato in quanto u_M(L,E) $= 1 \ngeq 2 = u_M$(L,M).
Osserviamo che trascurando le tabelle contenenti almeno un risultato B, la

funzione di utilità rispetto al tipo H è 50 volte l'utilità rispetto al tipo L.
Dunque: $u_M(\mathrm{L}, O_1) \geq u_M(\mathrm{L}, O_2)$
$u_M(\mathrm{L}, O_3) \geq u_M(\mathrm{L}, O_4)$
$u_M(\mathrm{H}, O_2) \geq u_M(\mathrm{H}, O_1) \implies 50u_M(\mathrm{L}, O_2) \geq 50u_M(\mathrm{L}, O_1)$
$u_M(\mathrm{H}, O_4) \geq u_M(\mathrm{H}, O_3) \implies 50u_M(\mathrm{L}, O_4) \geq 50u_M(\mathrm{L}, O_3)$

$\implies u_M(\mathrm{L}, O_1) = u_M(\mathrm{L}, O_2),\ \ u_M(\mathrm{L}, O_3) = u_M(\mathrm{L}, O_4)$
Analogamente: $u_F(\mathrm{L}, O_1) \geq u_F(\mathrm{L}, O_3)$
$u_F(\mathrm{L}, O_2) \geq u_F(\mathrm{L}, O_4)$
$u_F(\mathrm{H}, O_3) \geq u_F(\mathrm{H}, O_1) \implies 50u_F(\mathrm{L}, O_3) \geq 50u_F(\mathrm{L}, O_1)$
$u_F(\mathrm{H}, O_4) \geq u_F(\mathrm{H}, O_2) \implies 50u_F(\mathrm{L}, O_4) \geq 50u_F(\mathrm{L}, O_2)$

$\implies u_F(\mathrm{L}, O_1) = u_F(\mathrm{L}, O_3),\ \ u_F(\mathrm{L}, O_2) = u_F(\mathrm{L}, O_4)$

$\implies O_1 = O_2 = O_3 = O_4.$
L'implementazione nelle strategie dominanti impone dunque che gli unici meccanismi possibili siano quelli di tabella 2.5:

	M L	M H
F L	M	M
F H	M	M

	M L	M H
F L	F	F
F H	F	F

	M L	M H
F L	E	E
F H	E	E

Tabella 2.5: Tre soluzioni possibili per un meccanismo deterministico implementato secondo le strategie dominanti.

Valutiamo ora l'utilità attesa di ciasun agente. Nel primo caso (meccanismo con soli M) si ha:

$$\begin{aligned} u_M &= [0.8 \times u_M(L, M) + 0.2 \times u_M(L, M)] \times 0.8 + \\ &\quad [0.8 \times u_M(H, M) + 0.2 \times u_M(H, M)] \times 0.2 \\ &= [0.8 \times 2 + 0.2 \times 2] \times 0.8 + [0.8 \times 100 + 0.2 \times 100] \times 0.2 \\ &= 21.6 \end{aligned}$$

$$\begin{aligned} u_F &= [0.8 \times u_F(L, M) + 0.2 \times u_F(L, M)] \times 0.8 + \\ &\quad [0.8 \times u_F(H, M) + 0.2 \times u_F(H, M)] \times 0.2 \\ &= [0.8 \times 0 + 0.2 \times 0] \times 0.8 + [0.8 \times 0 + 0.2 \times 0] \times 0.2 \\ &= 0 \end{aligned}$$

Quindi si ha un'utilità totale di 21.6.

Nel secondo caso (meccanismo con soli F) si ha, data la simmetria della funzione di utilità: $u_M = 0, u_F = 21.6$ e quindi ancora un'utilità totale di 21.6.

Infine, per il meccanismo che ritorna sempre come risultato E (il quadro resta di entrambi) si ha:

$$\begin{aligned} u_M &= [0.8 \times u_M(L,E) + 0.2 \times u_M(L,E)] \times 0.8 + \\ &\quad [0.8 \times u_M(H,E) + 0.2 \times u_M(H,E)] \times 0.2 \\ &= [0.8 \times 1 + 0.2 \times 1] \times 0.8 + [0.8 \times 50 + 0.2 \times 50] \times 0.2 \\ &= 10.8 \end{aligned}$$

$$\begin{aligned} u_F &= [0.8 \times u_F(L,E) + 0.2 \times u_F(L,E)] \times 0.8 + \\ &\quad [0.8 \times u_F(H,E) + 0.2 \times u_F(H,E)] \times 0.2 \\ &= [0.8 \times 1 + 0.2 \times 1] \times 0.8 + [0.8 \times 50 + 0.2 \times 50] \times 0.2 \\ &= 10.8 \end{aligned}$$

con utilità totale 21.6.

Per quanto riguarda la funzione obiettivo queste tre soluzioni sono equivalenti. e quindi uno qualunque dei 3 meccanismi in Tabella 2.5 è accettabile.
Modifichiamo leggermente il problema richiedendo che l'implementazione avvenga secondo gli equilibri di Bayes-Nash.
Ora i due partecipanti non conoscono l'uno il tipo dell'altro. Il vincolo di partecipazione impone che l'utilità attesa di un'intera colonna in funzione del tipo corrispondente sia non negativa. Allo stesso modo l'utilità attesa di ogni riga, in funzione del tipo a capo di tale riga, deve essere non negativa.
L'implementazione negli equilibri di Bayes-Nash richiede poi che l'utilità attesa di ogni riga o colonna in funzione del tipo a cui si riferisce sia maggiore od uguale dell'utilità di ogni altra riga o colonna in funzione di quello stesso tipo. Per chiarire: supposto che l'agente M abbia tipo L, tenendo conto del fatto che non si conosce il tipo riportato dal secondo agente, dev'essere: $0.8 \times u_M(L,O_1) + 0.2 \times u_M(L,O_3) \geq 0.8 \times u_M(L,O_2) + 0.2 \times u_M(L,O_4)$.

Allora, scrivendo tutti i casi possibili per i due agenti:

$0.8 \times u_M(L, O_1) + 0.2 \times u_M(L, O_3) \geq 0$
$0.8 \times u_M(H, O_2) + 0.2 \times u_M(H, O_4) \geq 0$
$0.8 \times u_F(L, O_1) + 0.2 \times u_F(L, O_2) \geq 0$
$0.8 \times u_F(H, O_3) + 0.2 \times u_F(H, O_4) \geq 0$

$0.8 \times u_M(L, O_1) + 0.2 \times u_M(L, O_3) \geq$
$0.8 \times u_M(L, O_2) + 0.2 \times u_M(L, O_4)$

$0.8 \times u_M(H, O_2) + 0.2 \times u_M(H, O_4) \geq$
$0.8 \times u_M(H, O_1) + 0.2 \times u_M(H, O_3)$

$0.8 \times u_F(L, O_1) + 0.2 \times u_F(L, O_2) \geq$
$0.8 \times u_F(L, O_3) + 0.2 \times u_M(L, O_4)$

$0.8 \times u_F(H, O_3) + 0.2 \times u_F(H, O_4) \geq$
$0.8 \times u_F(H, O_1) + 0.2 \times u_F(H, O_2)$

In questo caso sono accettabili anche le tabelle contenenti come risultato B (bruciare il quadro), purchè l'utilità attesa su ciascuna riga o colonna sia non negativa. Poichè l'utilità di B è per entrambi -10 e la massima utilità in corrispondenza del tipo L è 2: $-10 + 2 \ngeq 0$, non si può avere come risultato B in nessuna cella in cui almeno uno dei due tipi sia L. Quindi: $O_1 \neq B, O_2 \neq B, O_3 \neq B$. (Altrimenti avrei un'utilità attesa non positiva). Sono sicuramente da scartare tutti i meccanismi che producano un risultato B al di fuori di o(H,H).
Cerchiamo per prima cosa le soluzioni non contenenti B come risultato. Osservando ancora che $u(H, O_i) = 50u(L, O_i)$ per ogni $O_i \neq B$, si ha:

$0.8 \times u_M(L, O_1) + 0.2 \times u_M(L, O_3) = 0.8 \times u_M(L, O_2) + 0.2 \times u_M(L, O_4)$
$0.8 \times u_F(L, O_1) + 0.2 \times u_F(L, O_2) = 0.8 \times u_F(L, O_3) + 0.2 \times u_M(L, O_4)$

Quindi tra le tabelle non contenenti B, le uniche accettabili sono quelle del caso precedente (tabella 2.5).

Il risultato B si può ottenere solo quando entrambi riportano il tipo H. Al-

lora:

$$0.8u_M(H, O_2) \geq 2 \quad \Longrightarrow O_2 = M, \text{ oppure } O_2 = E.$$
$$0.8u_F(H, O_3) \geq 2 \quad \Longrightarrow O_3 = F, \text{ oppure } O_3 = E.$$

Tra tutte le combinazioni possibili aventi $O_4 = B$, $O_3 \in \{E, F\}$, $O_2 \in \{E, M\}$, l'unica che soddisfi il vincolo di compatibilità è rappresentata in Tab. 2.6:

	M L	M H
F L	E	M
F H	F	B

Tabella 2.6: Meccanismo deterministico implementato negli equilibri di Bayes-Nash

La tabella 2.6 schematizza il meccanismo ottimo per questa particolare istanza del problema in quanto l'utilità totale attesa vale 32.48, ed è maggiore dell'utilità attesa delle tabelle precedenti. $u_{\text{tot}} = [0.8 \times u_M(L, E) + 0.2 \times u_M(L, F)] \times 0.8 +$
$[0.8 \times u_M(H, M) + 0.2 \times u_M(H, B)] \times 0.2 +$
$[0.8 \times u_F(L, E) + 0.2 \times u_F(L, M)] \times 0.8 +$
$[0.8 \times u_F(H, F) + 0.2 \times u_F(H, B)] \times 0.2$
$= [0.8 \times 1 + 0.2 \times 0] \times 0.8 + [0.8 \times 100 + 0.2 \times (-10)] \times 0.2 +$
$[0.8 \times 1 + 0.2 \times 0] \times 0.8 + [0.8 \times 100 + 0.2 \times (-10)] \times 0.2$
$= 32.48$

Allora, quando si rilassa il vincolo di compatibilità in BNE, si può in qualche caso avere una soluzione migliore bruciando il quadro.

Vediamo ora come migliorare i risultati permettendo la randomizzazione. Il meccanismo generato non fornisce, una volta selezionati i tipi, un risultato specifico, ma una distribuzione di probabilità su tutti i possibili risultati. Ad esempio: 0.5:M, 0.3:F, 0.2:B, 0:E significa che, con probabilità 0.5 il quadro verrà assegnato al marito, con probabilità 0.3 sarà assegnato alla moglie e con probabilità 0.2 verrà bruciato. Per semplificare la tabella non si scrivono i risultati che compaiono con probabilità nulla.

In ogni cella compare ora più di un risultato come mostra la tabella 2.7:

	M L	M H
F L	$[P_1 : O_1], [P_2 : O_2], [P_3 : O_3], [P_4 : O_4]$	$[P_5 : O_5], [P_6 : O_6], [P_7 : O_7], [P_8 : O_8]$
F H	$[P_9 : O_9], [P_{10} : O_{10}], [P_{11} : O_{11}], [P_{12} : O_{12}]$	$[P_{13} : O_{13}], [P_{14} : O_{14}], [P_{15} : O_{15}], [P_{16} : O_{16}]$

Tabella 2.7: Meccanismo randomizzato: in ogni cella possono comparire fino a 4 risultati. P_i indica la probabilità di verificarsi del risultato O_i.

Generiamo un meccanismo randomizzato che implementi i risultati negli equilibri di Bayes-Nash.
Imponiamo il vincolo di partecipazione:

$$0.8[u_M(L,O_1)P_1 + u_M(L,O_2)P_2 + u_M(L,O_3)P_3 + u_M(L,O_4)P_4] + 0.2[u_M(L,O_9)P_9 + u_M(L,O_{10})P_{10} + u_M(L,O_{11})P_{11} + u_M(L,O_{12})P_{12}] \geq 0$$

$$0.8[u_M(H,O_5)P_5 + u_M(H,O_6)P_6 + u_M(H,O_7)P_7 + u_M(H,O_8)P_8] + 0.2[u_M(H,O_{13})P_{13} + u_M(H,O_{14})P_{14} + u_M(H,O_{15})P_{15} + u_M(H,O_{16})P_{16}] \geq 0$$

$$0.8[u_F(L,O_1)P_1 + u_F(L,O_2)P_2 + u_F(L,O_3)P_3 + u_F(L,O_4)P_4] + 0.2[u_F(L,O_5)P_5 + u_F(L,O_6)P_6 + u_F(L,O_7)P_7 + u_F(L,O_8)P_8] \geq 0$$

$$0.8[u_F(H,O_9)P_9 + u_F(H,O_{10})P_{10} + u_F(H,O_{11})P_{11} + u_F(H,O_{12})P_{12}] + 0.2[u_F(H,O_{13})P_{13} + u_F(H,O_{14})P_{14} + u_F(H,O_{15})P_{15} + u_F(H,O_{16})P_{16}] \geq 0$$

Implementazione negli equilibri di Bayes-Nash:

$$0.8[u_M(L,O_1)P_1 + \cdots + u_M(L,O_4)P_4] + 0.2[u_M(L,O_9)P_9 + \cdots + u_M(L,O_{12})P_{12}] \geq 0.8[u_M(L,O_5)P_5 + \cdots + u_M(L,O_8)P_8] + 0.2[u_M(L,O_{13})P_{13} + \cdots + u_M(L,O_{16})P_{16}]$$

$$0.8[u_M(H,O_5)P_5 + \cdots + u_M(H,O_8)P_8] + 0.2[u_M(H,O_{13})P_{13} + \cdots + u_M(H,O_{16})P_{16}] \geq 0.8[u_M(H,O_1)P_1 + \cdots + u_M(H,O_4)P_4] + 0.2[u_M(H,O_9)P_9 + \cdots + u_M(H,O_{12})P_{12}]$$

$$0.8[u_F(L,O_1)P_1 + \cdots + u_F(L,O_4)P_4] + 0.2[u_F(L,O_5)P_5 + \cdots + u_F(L,O_8)P_8] \geq 0.8[u_F(L,O_9)P_9 + \cdots + u_F(L,O_{12})P_{12}] + 0.2[u_F(L,O_{13})P_{13} + \cdots + u_F(L,O_{16})P_{16}]$$

$0.8[u_F(H,O_9)P_9+\cdots+u_F(H,O_{12})P_{12}]+0.22[u_F(H,O_{13})P_{13}+\cdots+u_F(H,O_{16})P_{16}] \geq$
$0.8[u_F(H,O_1)P_1+\cdots+u_F(H,O_4)P_4]+0.2[u_F(H,O_5)P_5+\cdots+u_F(H,O_8)P_8]$

Trascurando i passaggi risolviamo questo problema di programmazione lineare che ha come soluzione ottima la tabella 2.8:

	M L	M H
F L	[0.57:M] , [0.43:F]	[1:M]
F H	[1:F]	[0.45:B] , [0.55:M]

Tabella 2.8: Meccanismo randomizzato che massimizza l'utilità attesa.

Calcolo l'utilità attesa:

$$\begin{aligned} u_M &= [0.8 \times u_M(L,M) \times 0.57 + 0.8 \times u_M(L,F) \times 0.43 + 0.2 \times u_M(L,F)] \times 0.8 + \\ &\quad [0.8 \times u_M(H,M) + 0.2 \times u_M(H,B) \times 0.45 + u_M(H,M) \times 0.55] \times 0.2 \\ &= [0.8 \times 2 \times 0.57 + 0.8 \times 0 \times 0.43 + 0.2 \times 0] \times 0.8 + \\ &\quad [0.8 \times 100 + 0.2 \times (-10) \times 0.45 + 100 \times 0.55] \times 0.2 \\ &= 0.73 + 16 - 0.18 + 2.2 = 16.37 \end{aligned}$$

$$\begin{aligned} u_F &= [0.8 \times u_F(L,M) \times 0.57 + 0.8 \times u_F(L,F) \times 0.43 + 0.2 \times u_F(L,M)] \times 0.8 + \\ &\quad [0.8 \times u_F(H,F) + 0.2 \times u_F(H,B) \times 0.45 + 0.2 \times u_F(H,M) \times 0.55] \times 0.2 \\ &= [0.8 \times 0 \times 0.57 + 0.8 \times 2 \times 0.43 + 0.2 \times 0] \times 0.8 + \\ &\quad [0.8 \times 100 + 0.2 \times (-10) \times 0.45 + 0.2 \times 0 \times 0.55] \times 0.2 \\ &= 0.55 + 16 - 1.18 = 16.37 \end{aligned}$$

$u_{\text{tot}} = 35.12$.
Quindi la randomizzazione permette di migliorare il risultato.

2.4.2 Un arbitro benevolo che usa i pagamenti

Si immagini ora di poter forzare i due partecipanti a pagare un prezzo. Siano quindi ammessi i pagamenti.
L'arbitro (per ora) si preoccupa solo del benessere dei due agenti, tenendo conto sia dell'assegnazione del quadro che dei soldi persi.

Implementazione nelle strategie dominanti:

$u_M(L, O_1) - \pi_M(L, L) \geq u_M(L, O_2) - \pi_M(H, L)$
$u_M(L, O_3) - \pi_M(L, H) \geq u_M(L, O_4) - \pi_M(H, H)$
$u_M(H, O_2) - \pi_M(H, L) \geq u_M(H, O_1) - \pi_M(L, L)$
$u_M(H, O_4) - \pi_M(H, H) \geq u_M(H, O_3) - \pi_M(L, H)$

$u_F(L, O_1) - \pi_F(L, L) \geq u_F(L, O_3) - \pi_F(L, H)$
$u_F(L, O_2) - \pi_F(H, L) \geq u_F(L, O_4) - \pi_F(H, H)$
$u_F(H, O_3) - \pi_F(L, H) \geq u_F(H, O_1) - \pi_F(L, L)$
$u_F(H, O_4) - \pi_F(H, H) \geq u_F(H, O_2) - \pi_F(H, L)$

Vincolo di partecipazione "ex post IR":

$u_M(L, O_1) - \pi_M(L, L) \geq 0$
$u_M(L, O_3) - \pi_M(L, H) \geq 0$
$u_M(H, O_2) - \pi_M(H, L) \geq 0$
$u_M(H, O_4) - \pi_M(H, H) \geq 0$

$u_F(L, O_1) - \pi_F(L, L) \geq 0$
$u_F(L, O_2) - \pi_F(H, L) \geq 0$
$u_F(H, O_3) - \pi_F(L, H) \geq 0$
$u_F(H, O_4) - \pi_F(H, H) \geq 0$

Abbiamo già visto come le uniche tabelle ammissibili contenenti il risultato B siano quelle per cui $O_1, O_2, O_3 \neq B$, inoltre, poichè deve valere il vincolo di partecipazione, devono essere soddisfatte le disuguaglianze:

$u_M(H, O_4) - \pi_M(H, H) \geq 0$
$u_F(H, O_4) - \pi_F(H, H) \geq 0.$

Necessariamente anche O_4 deve essere diverso da B.

Trascurando ancora una volta i passaggi, otteniamo come meccanismo deterministico ottimale per questa specifica istanza quello di Tab. 2.9:

	M L	M H
F L	M	M
F H	F	F

Tabella 2.9: Meccanismo deterministico implementato nelle strategie dominanti, in cui sono permessi i pagamenti come incentivo a riportare il proprio tipo in modo sincero.

E le funzioni di pagamento sono schematizzate in Tab. 2.10:

	M L	M H
F L	(0,0)	(0,0)
F H	(2,0)	(2,0)

Tabella 2.10: Tabella dei pagamenti:(i,j) indica che la moglie paga i e il marito paga j.

Calcolo l'utilità attesa, senza tenere conto dei pagamenti:

$$\begin{aligned} u_M &= [0.8 \times u_M(L,M) + 0.2 \times u_M(L,F)] \times 0.8 + \\ &\quad [0.8 \times u_M(H,M) + 0.2 \times u_M(H,F)] \times 0.2 \\ &= [0.8 \times 2 + 0.2 \times 0] \times 0.8 + [0.8 \times 100 + 0.2 \times 0] \times 0.2 \\ &= 1.28 + 16 = 17.28 \end{aligned}$$

$$\begin{aligned} u_F &= [0.8 \times u_F(L,M) + 0.2 \times u_F(L,M)] \times 0.8 + \\ &\quad [0.8 \times u_F(H,F) + 0.2 \times u_F(H,F)] \times 0.2 \\ &= [0.8 \times 0 + 0.2 \times 0] \times 0.8 + [0.8 \times 100 + 0.2 \times 100] \times 0.2 \\ &= 16 + 4 = 20 \end{aligned}$$

Con utilità totale 37.28.
La probabilità della moglie di pagare 2 è data da: $0.8 \times 0.2 + 0.2 \times 0.2 = 0.2$. Tale pagamento serve ad incentivare la moglie a non riportare falsamente il tipo H per aumentare la propria utilità.

2.4.3 Un arbitro interessato a massimizzare i pagamenti riscossi

Sia l'arbitro non benevolo ed interessato a massimizzare i pagamenti che gli pervengono dai due agenti. Naturalmente egli non può richiedere che gli venga pagata una somma qualunque; piuttosto deve fare in modo che le due parti siano ancora motivate a partecipare al meccanismo.
Richiediamo un vincolo ex post IR, e l'implementazione nelle strategie dominanti.

$$u_M(L, O_1) - \pi_M(L, L) \geq u_M(L, O_2) - \pi_M(H, L)$$
$$u_M(L, O_3) - \pi_M(L, H) \geq u_M(L, O_4) - \pi_M(H, H)$$
$$u_M(H, O_2) - \pi_M(H, L) \geq u_M(H, O_1) - \pi_M(L, L)$$
$$u_M(H, O_4) - \pi_M(H, H) \geq u_M(H, O_3) - \pi_M(L, H)$$

$$u_F(L, O_1) - \pi_F(L, L) \geq u_F(L, O_3) - \pi_F(L, H)$$
$$u_F(L, O_2) - \pi_F(H, L) \geq u_F(L, O_4) - \pi_F(H, H)$$
$$u_F(H, O_3) - \pi_F(L, H) \geq u_F(H, O_1) - \pi_F(L, L)$$
$$u_F(H, O_4) - \pi_F(H, H) \geq u_F(H, O_2) - \pi_F(H, L)$$

$$u_M(L, O_1) - \pi_M(L, L) \geq 0$$
$$u_M(L, O_3) - \pi_M(L, H) \geq 0$$
$$u_M(H, O_2) - \pi_M(H, L) \geq 0$$
$$u_M(H, O_4) - \pi_M(H, H) \geq 0$$

$$u_F(L, O_1) - \pi_F(L, L) \geq 0$$
$$u_F(L, O_2) - \pi_F(H, L) \geq 0$$
$$u_F(H, O_3) - \pi_F(L, H) \geq 0$$
$$u_F(H, O_4) - \pi_F(H, H) \geq 0$$

Le condizioni imposte dai vincoli sono le stesse del caso precedente, ma cambia la funzione obiettivo: non si massimizza più il benessere sociale, ma i pagamenti riscossi.
Poichè deve essere

$u_M(t_i, o(t_i, t_j)) - \pi_M(t_i, t_j) \geq 0$ e
$u_F(t_j, o(t_i, t_j)) - \pi_F(t_i, t_j) \geq 0$

come funzione di pagamento l'arbitro può richiedere, ad esempio, l'intera utilità degli agenti corrispondente all'assegnazione. Per fare ciò bisogna tener conto anche di eventuali pagamenti negativi (risarcimenti): per poter massimizzare la funzione di utilità (e di conseguenza i pagamenti) di ciascun agente, rispettando il vincolo IC, bisogna bruciare il quadro quando entrambi riportano uno scarso interesse. L'arbitro deve compensare le parti in causa per la perdita del dipinto. Si ottiene il meccanismo di Tab. 2.11:

	M L	M H
F L	B	M
F H	F	F

Tabella 2.11: Arbitro interessato a massimizzare i pagamenti

Con tabella dei pagamenti Tab. 2.12:

	M L	M H
F L	(-10,-10)	(0,100)
F H	(100,0)	(100,0)

Tabella 2.12: Massimizzazione dei pagamenti.(i,j) indica che la moglie paga i e il marito paga j. Un pagamento negativo rappresenta un risarcimento da parte dell'arbitro.

Capitolo 3

Problemi vincolati multi-agente

Nei capitoli precedenti sono stati presentati due importanti argomenti della letteratura informatica, nell'ambito dell'intelligenza artificiale: i problemi di soddisfazione di vincoli e la progettazione automatizzata di meccanismi.
Mentre da un lato l'introduzione del concetto di vincoli soft amplia il campo di applicabilità della programmazione con vincoli, permettendo di assegnare delle preferenze, dall'altro sorge il problema di gestire tali valutazioni, qualora siano riportate in maniera distinta da un gruppo di agenti.
In questo capitolo vediamo come adattare il modello AMD proposto da Sandholm e Conitzer allo scopo di risolvere problemi multi-agente con vincoli soft.

Sia dato un insieme $V = \{V_1, V_2, \ldots, V_k\}$ di k variabili, aventi uno stesso dominio D di n valori possibili.
Un'assegnazione di valori alle variabili consiste in una k-upla[1]:$(a_1, \ldots, a_k)$ in cui $a_i \in D$ è il valore assegnato alla variabile V_i.
Vi siano poi N agenti: $A_1, \ldots, A_N$ interessati a dare ciascuno una propria valutazione sulle possibili assegnazioni. Come nella progettazione automatizzata di un meccanismo vista al capitolo 2, ogni agente riporta il proprio grado di interesse dichiarando un tipo ed un valore di utilità (o preferenza).
Supponiamo che gli N agenti abbiano lo stesso insieme T di tipi e che ogni tipo $t_j \in T$ rappresenti un ben definito intervallo di valori di utilità. Sia dunque $R_i = T \times \mathbb{R}$, prodotto cartesiano tra l'insieme T di tipi e l'insieme $\mathbb{R}$ dei numeri reali. I suoi elementi saranno le coppie: $< t_i, v_i >$ con

[1] n-upla di dimensione k.

$v_i = u_i(t_i, (a_1, \ldots, a_k))$ utilità dell'agente A_i in relazione all'assegnazione di valori $(V_1, \ldots, V_k) = (a_1, \ldots, a_k)$ e al tipo t_i, e tali che:

$$\forall < t_1, v_1 >, < t_2, v_2 > \in R_i, \begin{cases} t_1 \leq t_2 & \Leftrightarrow & v_1 \leq v_2 \\ t_1 \neq t_2 & \Rightarrow & v_1 \neq v_2 \end{cases}$$

La mancanza di simmetria in queste due condizioni è dovuta al fatto che, per ogni agente, un tipo $\tilde{t}$ non corrisponde ad un determinato valore numerico, ma ad un intervallo di valori. Ogni valutazione numerica, invece, è associata ad un unico tipo, essendo disgiunti gli intervalli corrispondenti a tipi distinti.

Un esempio:
T={L,H} con $\begin{cases} L & \sim & [0.1 \ldots 0.5] \\ H & \sim & [0.6 \ldots 1.0] \end{cases}$
Se consideriamo $< L, 0.2 >$ e $< L, 0.5 >$ si ha:
$0.2 \leq 0.5 \Rightarrow L \leq L$, ma $0.1 \neq 0.5 \nRightarrow L \neq L$!

Si può osservare come sugli elementi di R_i valga una relazione di ordine parziale $\leq_R$ indotta dall'ordinamento dei reali:
$\forall < t_1, v_1 >, < t_2, v_2 > \in R_i, < t_1, v_1 > \leq_R < t_2, v_2 > \Leftrightarrow v_1 \leq v_2$.

Ogni agente associa, ad un insieme di assegnazioni di valori per coppie di variabili, un elemento di R_i. Tale elemento rappresenta la preferenza dell'agente relativa a quell'assegnazione.
Definiamo ora due operazioni su R_i che permettono di confrontare tra loro le valutazioni degli agenti e di combinarle per ottenere le preferenze riguardanti tuple complete di assegnazioni.

• Sia $+$: $R_i \times R_i \longrightarrow R_i$ tale che $< t_1, v_1 > + < t_2, v_2 > = < t_2, v_2 > \Leftrightarrow < t_1, v_1 > \leq_R < t_2, v_2 >$.
Esiste $< \hat{t}, \hat{v} >$ tale che $< \hat{t}, \hat{v} > + < t, v > = < t, v > = < t, v > + < \hat{t}, \hat{v} > \forall < t, v > \in R_i$
($< \hat{t}, \hat{v} >$ è l'elemento neutro dell'addizione).
Allora $< \hat{t}, \hat{v} >$ è il più piccolo elemento di R_i rispetto all'ordine $\leq_R$.

• Sia $\times$ un'operazione moltiplicativa tale che $\forall (a_1, \ldots, a_k) \in D^k$ si ha
$u_i(\bar{t}, (a_1, \ldots, a_k)) = Val_1 \times Val_2 \times \ldots \times Val_r$
(Val_j è la valutazione rispetto a $\bar{t}$ dell'j-esima assegnazione che compone il

risultato[2]. $r \leq \frac{k(k-1)}{2}$ perchè alcune valutazioni possono non essere state considerate.).

Non necessariamente tutti gli agenti devono combinare allo stesso modo le proprie valutazioni, l'importante è che venga rispettato l'ordinamento definito sopra.

Si può facilmente osservare che, con le operazioni appena descritte, R_i è un semianello:

$$< R_i = T \times \mathbb{R}, +, \times, < \hat{t}, \hat{v} >= min\{< t, v > \in R_i\}, \bar{1} >$$

Infatti:

- R_i è un dominio contenente gli elementi $< \hat{t}, \hat{v} >$ e $\bar{1}$;
- $+$ è un'operazione additiva, commutativa, associativa e avente $< \hat{t}, \hat{v} >$ come elemento neutro;
- $\times$ è un'operazione moltiplicativa, associativa, avente $\bar{1}$ come elemento neutro e $< \hat{t}, \hat{v} >$ come elemento assorbente.

Una caratteristica del modello che stiamo costruendo è che, poichè la distribuzione di probabilità con cui ogni agente riporta i tipi non è costante, ma varia in base alla particolare assegnazione che si sta valutando, le dichiarazioni consistono in triple: $< P_t(a), t, v(a) >$ dove $P_t(a)$ è la probabilità del tipo t relativamente all'assegnazione a, mentre $v(a)$ è la valutazione legata al tipo t dell'utilità dell'assegnazione a.

La principale differenza, rispetto al modello AMD descritto al capitolo precedente, consiste nel fatto che, nonostante i risultati siano le assegnazioni complete di valori a tutte le variabili del problema con vincoli, le preferenze degli agenti sono espresse, tramite i vincoli, su piccoli sottoinsimi di tali varibili. Si suppone, infatti, che le valutazioni degli agenti non siano riportate

[2]Se un agente non esprime alcuna valutazione su qualche coppia di variabili, nella composizione viene considerato l'elemento neutro $\bar{1}$ dell'operazione moltiplicativa appena definita (dipende dalla particolare operazione $\times$ scelta).

sulle assegnazioni complete di valori, ma sulle alcune delle coppie che le compongono.

Sia ad esempio:
$V = \{X, Y, Z\}$,
$D = \{0, 1\}$
$T = \{L, H\}$, con $L = [1, \ldots, 5]$; $H = [6, \ldots, 10]$.
Le terne possibili nell'insieme con 2 elementi {0,1} sono:
(000); (001); (010); (011); (100); (101); (110); (111).
Ogni agente valuta singolarmente le coppie di valori che possono essere assegnate a (X,Y),(X,Z) e (Y,Z).
Il coordinatore calcola le valutazioni riguardanti la terna (X,Y,Z) componendo quelle riportate mediante l'operatore $\times$.
In questo modo ciascun agente è libero di valutare una parte o tutte le coppie di variabili.
Quando si compongono le valutazioni, alle coppie non considerate viene associato l'elemento neutro dell'operatore di combinazione.

Supponiamo che le due tabelle 3 schematizzino le valutazioni riportate da un agente A su un sistema con tre variabili.

(X,Y)		
(0,0)	0.5 L 2	0.5 H 8
(0,1)	0.5 L 4	0.5 H 9
(1,0)	0.5 L 1	0.5 H 7
(1,1)	0.5 L 2	0.5 H 6

(Y,Z)		
(0,0)	0.5 L 1	0.5 H 9
(0,1)	0.5 L 4	0.5 H 6
(1,0)	0.5 L 3	0.5 H 6
(1,1)	0.5 L 3	0.5 H 8

Tabella 3.1: Valutazioni riportate da un agente su un sitema con tre variabili.

E sia $\times$=min l'operatore dicombinazione. Si ottengono le valutazioni sulle tuple complete di Tab. 3.2.
Poichè l'agente non considera la coppia (X,Z), ad essa viene assegnata per default la valutazione di tabella 3.
perchè ogni valutazione legata al tipo L ha un valore $v \leq 5$ e quindi $min\{v, 5\} = v$, e ogni valutazione legata al tipo H ha un valore $w \leq 10$ e quindi $min\{w, 10\} = w$.

La probabilità che il tipo relativo alla valutazione dell'agente A_i sull'assegnazione

(X,Y,Z)		
(0,0,0)	0.5 L 1	0.5 H 8
(0,0,1)	0.5 L 2	0.5 H 6
(0,1,0)	0.5 L 3	0.5 H 6
(0,1,1)	0.5 L 3	0.5 H 8
(1,0,0)	0.5 L 3	0.5 H 7
(1,0,1)	0.5 L 1	0.5 H 6
(1,1,0)	0.5 L 2	0.5 H 6
(1,1,1)	0.5 L 2	0.5 H 6

Tabella 3.2: Valutazioni complete ottenute combinando i valori delle tabelle 3 calcolando il minimo.

(X,Z)		
(0,0)	0.5 L 5	0.5 H 10
(0,1)	0.5 L 5	0.5 H 10
(1,0)	0.5 L 5	0.5 H 10
(1,1)	0.5 L 5	0.5 H 10

Tabella 3.3: L'agente valuta due coppie di variabili su tre. Alla terza coppia viene assegnato l'elemento neutro dell'operatore di combinazione.

completa $(\bar{a_1}, \ldots, \bar{a_k})$ sia proprio $\bar{t}$ si ottiene componendo le probabilità dello stesso tipo nelle varie coppie $(\bar{a_i}, \bar{a_j})$ che generano l'assegnazione.

Un breve esempio: sia
$V = \{X, Y, Z\}$,
$D = \{0, 1\}$,
$T = \{L, H\}$.
Per l'agente A si abbiano le valutazioni di tabella 3

(X,Z)		
(0,0)	[0.45 L 5]	[0.55 H 10]
(0,1)	[0.55 L 5]	[0.45 H 10]
(1,0)	[0.35 L 5]	[0.65 H 10]
(1,1)	[0.60 L 5]	[0.40 H 10]

(Y,Z)		
(0,0)	[0.25 L 5]	[0.75 H 10]
(0,1)	[0.30 L 5]	[0.70 H 10]
(1,0)	[0.55 L 5]	[0.45 H 10]
(1,1)	[0.60 L 5]	[0.40 H 10]

Tabella 3.4: Valutazioni riportate da un agente su un sitema con tre variabili.

Un'operazione che permette di combinare le probabilità mantenendo la complementarietà: $P(t = L|O) = 1 - P(t = H|O)$ è la media aritmetica approssimata alla seconda cifra dopo la virgola. I risultati ottenuti sono schematizzati in Tab. 5.2.

(X,Y,Z)	
(0,0,0)	[0.35 L 5] [0.65 H 10]
(0,0,1)	[0.38 L 5] [0.62 H 10]
(0,1,0)	[0.55 L 5] [0.45 H 10]
(0,1,1)	[0.58 L 5] [0.42 H 10]
(1,0,0)	[0.30 L 5] [0.70 H 10]
(1,0,1)	[0.33 L 5] [0.67 H 10]
(1,1,0)	[0.58 L 5] [0.42 H 10]
(1,1,1)	[0.60 L 5] [0.40 H 10]

Tabella 3.5: Le probabilità in tabella 3 vengono combinate tramite la media aritmetica.

Altra importante differenza rispetto al modello AMD è che gli agenti non definiscono i loro tipi in funzione di un particolare risultato; nell'esempio 2.4 a pagina 25 i due tipi L e H rappresentavano l'interesse che i due agenti avevano per il bene in questione, e dunque il particolare interesse per uno dei 4 risultati possibili: ricevere il quadro. A partire da questo, poi, venivano valutate le altre 3 alternative.
Nel nostro caso non vi è alcun assegnamento preferibile, rispetto al quale giudicare le alternative.

3.1 La progettazione del meccanismo

Una volta definiti i dati del problema, cioè dopo aver combinato tra loro le valutazioni, si può procedere con la progettazione del meccanismo.
Anche in queste particolari istanze gli agenti devono essere motivati a partecipare. ed a riportare il proprio tipo veritiero. Vediamo, allora, la formulazione dei vincoli IR ed IC, adattata al nostro problema.

A differenza dei vincoli descritti a pag. 21 e seguenti, nel nostro caso non vengono presi in considerazione i pagamenti, quindi l'utilità di ciascun agente dipende unicamente dal risultato e dal tipo a cui si riferisce. Inoltre, in questa nostra rappresentazione bisogna tener conto del fatto che anche le probabilità dei tipi variano variano a seconda del particolare risultato che si sta considerando, come visto a pag. 41 . Scriviamo come $P(t_i = \bar{t} \mid O_j)$ la probabilità che il tipo t_i dell'agente A_i sia $\bar{t}$ quando si valuta il risultato O_j.

Perchè sia rispettato il vincolo "ex interim IR" nella progettazione del meccanismo, il valore atteso per ogni agente, in corrispondenza di ogni suo tipo $\bar{t_i}$ e al variare delle possibili valutazioni degli altri agenti, deve essere non negativo. Quindi, per ogni agente A_i ed ogni tipo $\bar{t_i} \in T$ dev'essere:

(D)

$$\sum_{(\bar{t}_1,\ldots,\bar{t}_{i-1},\bar{t}_{i+1},\ldots,\bar{t}_N)|\bar{t}_i} \Big[\prod_{\substack{j=1 \\ j\neq i}}^{N} P(t_j = \bar{t}_j \mid o(\bar{t}_1,\ldots,\bar{t}_N) \times u_i(\bar{t}_i, o(\bar{t}_1,\ldots,\bar{t}_N)) \Big] \geq 0$$

(R)

$$\sum_{(\bar{t}_1,\ldots,\bar{t}_{i-1},\bar{t}_{i+1},\ldots,\bar{t}_N)|\bar{t}_i} \Big[\sum_{\bar{O}|(\bar{t}_1,\ldots,\bar{t}_N)} P(\bar{O}) \times \prod_{\substack{j=1 \\ j\neq i}}^{N} P(t_j = \bar{t}_j \mid \bar{O}) \times u_i(\bar{t}_i, \bar{O}) \Big] \geq 0$$

Se, invece, si richiede sia rispettato il vincolo di partecipazione a posteriori ("ex post IR"), è sufficiente che sia non negativa l'utilità (dell'agente A_i in funzione del suo tipo t_i) del risultato corrispondente ad ogni ben definito vettore di tipi $(t_1 \ldots t_i \ldots t_N)$.
Allora, per ogni agente A_i e per ogni vettore di tipi $(t_1 \ldots t_N) \in T^N$ dev'essere:

(D)

$$u_i(t_i, o(t_1,\ldots,t_N)) \geq 0$$

(R)

$$\sum_{o(t_1,\ldots,t_N)} P(o(t_1,\ldots,t_N)) \times u_i(t_i, o(t_1,\ldots,t_N)) \geq 0$$

((D) ed (R) indicano rispettivamente un meccanismo deterministico ed un meccanismo randomizzato.)
Bisogna ricordare che, a differenza di quello deterministico, un meccanismo randomizzato non seleziona un particolare risultato, bensì una distribuzione di probabilità su tutti i risultati possibili.

Anche i due concetti di soluzione rimangono invariati, salvo per il fatto che le probabilità sui tipi dipendono dal risultato.
Si ha un'implementazione nelle strategie dominanti se per ogni agente A_i ed ogni vettore di tipi $(t_1 \ldots t_i \ldots t_N) \in T_1 \times \ldots \times T_i \times \ldots \times T_N$ l'utilità dell'agente considerato in funzione del suo tipo veritiero è maggiore dell'utilità che egli otterrebbe dichiarando un qualsiasi altro tipo $\hat{t}_i$:

(D)

$$u_i(t_i, o(t_1 \ldots t_i \ldots t_N)) \geq u_i(t_i, o(t_1 \ldots \hat{t}_i \ldots t_N))$$

(R)

$$\sum_{O=o(t_1,\ldots,t_i,\ldots,t_N)} P(O) \times u_i(t_i, O) \geq \sum_{\hat{O}=o(t_1,\ldots,\hat{t}_i,\ldots,t_N)} P(\hat{O}) \times u_i(t_i, \hat{O})$$

Si ha un'implementazione negli equilibri di Bayes-Nash se, pur non conoscendo i tipi effettivi degli altri agenti, ogni partecipante ottiene la massima utilità riportando il proprio tipo veritiero.
Per ogni agente A_i, per ogni tipo $t_i \in T$ ed ogni tipo alternativo $\hat{t}_i$ dev'essere:

(D)

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\prod_{\substack{j=1 \\ j\neq i}}^{N} P(t_j \mid o(t_1,\ldots,t_i,\ldots,t_N)) \times u_i(t_i, o(t_1,\ldots,t_i,\ldots,t_N)) \Big] \geq$$

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\prod_{\substack{j=1\\ j\neq i}}^{N} P(t_j \mid o(t_1,\ldots,\hat{t}_i,\ldots,t_N)) \times u_i(t_i, o(t_1,\ldots,\hat{t}_i,\ldots,t_N))\Big]$$

(R)

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\sum_{o|(t_1,\ldots,t_N)} P(\bar{O} = o(t_1,\ldots,t_i,\ldots,t_N)) \times \prod_{\substack{j=1\\ j\neq i}}^{N} P(t_j \mid \bar{O}) \times u_i(t_i,\bar{O})\Big] \geq$$

$$\sum_{(t_1,\ldots,t_{i-1},t_{i+1},\ldots,t_N)|t_i} \Big[\sum_{\hat{O}|(t_1,\ldots,\hat{t}_i,\ldots,t_N)} P(\hat{O} = o(t_1,\ldots,\hat{t}_i,\ldots,t_N)) \times \prod_{\substack{j=1\\ j\neq i}}^{N} P(t_j \mid \hat{O}) \times u_i(t_i,\hat{O})\Big]$$

3.1.1 Rappresentazioni alternative

Se noi combiniamo le valutazioni come detto sopra, per ogni assegnazione completa abbiamo una tripla $< P, t, v >$ per ogni tipo $t \in T$. Il meccanismo che poi viene costruito può essere schematizzato con una tabella a N dimensioni e $|T|$ entrate per ogni agente.
Una rappresentazione alternativa si ha considerando ogni coppia di valori come un singolo oggetto su cui riportare le valutazioni. Per ogni assegnazione completa non vi saranno più $|T|$ tipi, ma tutte le $|T|^{\frac{|V|(|V|-1)}{2}}$ combinazioni possibili di tipi.
Si ottiene una complicazione nella progettazione del meccanismo dovuta sia alle grandi dimensioni della tabella, sia al fatto che bisogna tenere in considerazione anche tutte le diverse probabilità dei tipi. Inoltre in questo caso ciascun agente dovrebbe esaminare necessariamente ogni coppia di variabili,

non essendoci un tipo preferibile da assegnare come default.

Esempio: anche se si hanno solo due tipi, $T = \{L, H\}$, si ottengono tutte le seguenti combinazioni: LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH. Sono $|T|^{\frac{|V|(|V|-1)}{2}}$ valutazioni diverse. Bisogna quindi calcolare $|T|^{\frac{|V|(|V|-1)}{2}}$ probabilità e $|T|^{\frac{|V|(|V|-1)}{2}}$ utilità per ogni assegnazione finale. Le assegnazioni possibili sono $|D|^{|V|}$ quindi ci sono $2|T|^{\frac{|V|(|V|-1)}{2}}|D|^{|V|}$ valori da calcolare.

Un altro problema si ha nel combinare le utilità relative a tipi diversi:
se L ed H rappresentano due intervalli disgiunti di preferenza, combinando tra loro valori corrispondenti a valutazioni diverse, non è più possibile rispettare la relazione d'ordine precedentemente descritta.
Consideriamo ad esempio $L = [1, 50]$, $H = [51, 100]$.
$\forall < t_1, v_1 >, < t_2, v_2 >: < t_1, v_1 > \leq_R < t_2, v_2 > \Leftrightarrow v_1 \leq v_2$.
Sembrerebbe logico porre $LLH \leq HLH$ Ma se il valore di LLH si ottiene combinando rispettivamente $48,\ 49,\ 100$ mediante media aritmetica si ottiene: $LLH \sim 65.67$.
Analogamente, se HLH si ottiene da $51,\ 49,\ 51$ si ha: $HLH \sim 50.33$.
Secondo la definizione dell'ordinamento $\leq_R$ data a pagina 40 : $< t_1, v_1 > \leq_R < t_2, v_2 > \Leftrightarrow v_1 \leq v_2$, si avrebbe: $HLH \leq LLH$.
In una diversa valutazione, invece, si potrebbe avere per $LLH \sim \frac{20+55+27}{3} = 34$, mentre $HLH \sim \frac{88+40+91}{3} = 73$, quindi $LLH \leq HLH$. L'ordinamento $\leq_R$ non sarebbe così ben definito.

Nel capitolo 2 il meccanismo automatizzato viene costruito prima di conoscere definitivamente i tipi degli agenti: vengono considerati tutti i possibili vettori di tipi $(t_1, \ldots, t_N) \in T^N$ e per ciascuno di essi viene selezionato un risultato soddisfacente le particolari condizioni imposte dall'istanza specifica del problema.
Il fatto di avere un elenco dei possibili risultati in funzione dei tipi dovrebbe servire agli agenti per verificare che i vincoli di razionalità individuale e di compatibilità sono stati rispettati nella progettazione del meccanismo. Tuttavia questo approccio risulta inefficiente quando la tabella che si dovrebbe produrre è troppo grande.
Nel modello che stiamo costruendo supponiamo di poter generare solo i risultati

riguardanti i tipi effettivamente dichiarati, garantendo un meccanismo prodotto secondo l'implementazione richiesta, ma senza doverlo dimostrare. (Negli esempi verranno prodotte le tabelle con lo scopo di illustrare il procedimento e le considerazioni che vi sono state fatte.)

3.2 Esempio

Consideriamo il semplice problema in cui sono date tre variabili $V = \{X, Y, Z\}$ con uno stesso dominio $D = \{1, 2\}$.
Due agenti: A_1, A_2 riportano alcune valutazioni sui possibili assegnamenti di valori a coppie di variabili, riferendo "un tipo" e una valutazione numerica. Sia $T = \{L, H\}$ tale che: $L = [1, 50]$ $H = [51, 100]$ e le valutazioni dei due agenti siano quelle di tabella 3.6 e 3.7.

(X,Y)			
(1,1)	0.3	L	30
	0.7	H	80
(1,2)	0.2	L	46
	0.8	H	91
(2,1)	0.4	L	12
	0.6	H	75
(2,2)	0.5	L	29
	0.5	H	61

(Y,Z)			
(1,1)	0.6	L	25
	0.4	H	93
(1,2)	0.3	L	9
	0.7	H	52
(2,1)	0.2	L	34
	0.8	H	60
(2,2)	0.7	L	23
	0.3	H	81

(X,Z)			
(1,1)	0.7	L	29
	0.3	H	73
(1,2)	0.2	L	15
	0.8	H	90
(2,1)	0.5	L	48
	0.5	H	88
(2,2)	0.6	L	44
	0.4	H	70

Tabella 3.6: Valutazioni del primo agente

Gli agenti non valutano globalmente le assegnazioni complete, ma riportano il loro giudizio sui valori riferiti a coppie di variabili. Definiscono poi un'operazione moltiplicativa $\times$ che permette di combinare tali valutazioni.

Supponiamo che per quanto riguarda il problema in questione sia:
$\times_1$ =valore intero della media aritmetica.
$\times_2$ =il minimo tra i livelli di preferenza.
Calcoliamo allora l'utilità (in corrispondenza dei due tipi) di ogni assegnazione completa. Poichè la probabilità con cui ogni agente riporta ciascun tipo non è costante ma dipende dalla particolare assegnazione che si sta considerando, la probabilità di ciascun tipo nella soluzione finale dipende dalla soluzione stessa. Per combinare tra loro le probabilità si può calcolare, ad esempio, la media aritmetica, come in Tab. 3.8.

(X,Y)			
(1,1)	0.5	L	21
	0.5	H	70
(1,2)	0.4	L	33
	0.6	H	84
(2,1)	0.6	L	49
	0.4	H	78
(2,2)	0.5	L	12
	0.5	H	60

(Y,Z)			
(1,1)	0.5	L	12
	0.5	H	85
(1,2)	0.5	L	6
	0.5	H	51
(2,1)	0.5	L	48
	0.5	H	53
(2,2)	0.5	L	24
	0.5	H	66

(X,Z)			
(1,1)	0.6	L	7
	0.4	H	55
(1,2)	0.5	L	39
	0.5	H	61
(2,1)	0.6	L	20
	0.4	H	55
(2,2)	0.4	L	12
	0.6	H	80

Tabella 3.7: Valutazioni del secondo agente

(X,Y,Z)	Agente A_1			Agente A_2		
(1,1,1)	0.53	L	28	0.53	L	7
	0.47	H	82	0.47	H	55
(1,1,2)	0.27	L	18	0.50	L	6
	0.73	H	74	0.50	H	51
(1,2,1)	0.37	L	36	0.40	L	7
	0.63	H	74	0.60	H	53
(1,2,2)	0.37	L	28	0.47	L	7
	0.63	H	87	0.53	H	61
(2,1,1)	0.50	L	28	0.57	L	6
	0.50	H	85	0.43	H	55
(2,1,2)	0.43	L	21	0.50	L	6
	0.57	H	65	0.50	H	51
(2,2,1)	0.40	L	37	0.53	L	12
	0.60	H	69	0.47	H	53
(2,2,2)	0.60	L	32	0.47	L	12
	0.40	H	68	0.53	H	60

Tabella 3.8: Valutazioni dei due agenti, calcolate sulle assegnazioni complete

Costruiamo un meccanismo deterministico che si può schematizzare con la tabella 3.9:
L'utilità attesa totale di una particolare soluzione sarà data da:

$$U_{TOT} = P(t_1 = L, t_2 = L|o(LL) = O_1) \times [u_1(L, O_1) + u_2(L, O_1)] + \\ P(t_1 = H, t_2 = L|o(HL) = O_2) \times [u_1(H, O_2) + u_2(L, O_2)] +$$

	L	H
L	o(LL)	o(HL)
H	o(LH)	o(HH)

Tabella 3.9: Meccanismo deterministico generico per il problema di assegnazione.

$$P(t_1 = L, t_2 = H|o(LH) = O_3) \times [u_1(L, O_3) + u_2(H, O_3)] +$$
$$P(t_1 = H, t_2 = H|o(HH) = O_4) \times [u_1(H, O_4) + u_2(H, O_4)]$$

Ad esempio, per la tabella 3.2:

	L	H
L	(221)	(121)
H	(122)	(122)

si ha:

$$U_{TOT} = 0.4 \times 0.53 \times [37 + 12] + 0.63 \times 0.4 \times [74 + 4] +$$
$$0.37 \times 0.53 \times [28 + 61] + 0.47 \times 0.53 \times [24 + 41]$$
$$69.47.$$

Meccanismo deterministico, ex post IR, implementato nelle strategie dominanti

Dato il problema descritto sopra, vogliamo costruire un meccanismo che permetta di trovare un'assegnazione di valori alle variabili, in modo da soddisfare il più possibile i due agenti.
Supponiamo che ogni agente conosca il tipo riportato dalla controparte.
Imponiamo il vincolo di partecipazione ex post IR:

$$u_1(L, O_1) \geq 0 \qquad u_2(L, O_1) \geq 0$$
$$u_1(L, O_3) \geq 0 \qquad u_2(H, O_3) \geq 0$$
$$u_1(H, O_2) \geq 0 \qquad u_2(L, O_2) \geq 0$$
$$u_1(H, O_4) \geq 0 \qquad u_2(H, O_4) \geq 0$$

Poichè tutte le valutazioni degli agenti sono positive queste condizioni sono banalmente soddisfatte. Tra tutte le soluzioni possibili, ci interessa quella che massimizza l'utilità attesa.

Cerchiamo O_1 che massimizzi $P(t_1 = L, t_2 = L|o(LL) = O_1) \times [u_1(L, O_1) + u_2(L, O_1)]$.

Si ottiene: $O_1 = (222)$.
Allo stesso modo si ottiene: $O_2 = (112)$, $O_3 = (222)$, $O_4 = (122)$

Quindi la tabella 3.2:

	L	H
L	(222)	(112)
H	(222)	(122)

Tabella 3.10: Meccanismo deterministico ottimo senza vincolo di compatibilità. L'utilità attesa di questa tabella vale 121.283 e può essere considerata come un limite superiore per l'utilità attesa che si può ottenere, implementando in qualsiasi modo il meccanismo.

Il meccanismo prodotto è deterministico e ottimo; incentiva gli agenti a partecipare, ma non a riportare il proprio vero tipo.
Infatti, poichè $u_2(L, O_2) \ngeq u_2(L, O_4)$, il secondo agente ottiene un'utilità maggiore mentendo quando $t_1 = H$ e $t_2 = L$.
Richiediamo allora che gli agenti siano incentivati a dire il vero:

$u_1(L, O_1) \geq u_1(L, O_2)$ $\quad 32 \geq 18$ $\surd$
$u_1(L, O_3) \geq u_1(L, O_4)$ $\quad 32 \geq 28$ $\surd$
$u_1(H, O_2) \geq u_1(H, O_1)$ $\quad 74 \geq 68$ $\surd$
$u_1(H, O_4) \geq u_1(H, O_3)$ $\quad 87 \geq 68$ $\surd$

$u_2(L, O_1) \geq u_2(L, O_3)$ $\quad 12 \geq 12$ $\surd$
$u_2(L, O_2) \geq u_2(L, O_4)$ $\quad 06 \geq 07$ NO!
$u_2(H, O_3) \geq u_2(H, O_1)$ $\quad 60 \geq 60$ $\surd$
$u_2(H, O_4) \geq u_2(H, O_2)$ $\quad 61 \geq 51$ $\surd$

Per ottenere una tabella che soddisfi le condizioni scritte sopra possiamo modificare O_2 oppure O_4. Tra le varie soluzioni scelgiamo quella che mi permette di ottenere un'utilità attesa maggiore.

Primo caso: modifico O_2. Deve valere:

$u_1(L, O_1) \geq u_1(L, O_2)$
$u_1(H, O_2) \geq u_1(H, O_1)$
$u_2(L, O_2) \geq u_2(L, O_4)$
$u_2(H, O_4) \geq u_2(H, O_2)$

Le soluzioni ammissibili sono:

$O_2 = (111)$ $\quad U_{TOT}(O_2 = (111)) = 114.253$
$O_2 = (122)$ $\quad U_{TOT}(O_2 = (122)) = 119.916$
$O_2 = (222)$ $\quad U_{TOT}(O_2 = (222)) = 107.123$

SECONDO CASO: modifico O_4. Deve valere:

$$u_1(L, O_3) \geq u_1(L, O_4)$$
$$u_1(H, O_4) \geq u_1(H, O_3)$$
$$u_2(L, O_2) \geq u_2(L, O_4)$$
$$u_2(H, O_4) \geq u_2(H, O_2)$$

Le soluzioni ammissibili sono:

$$O_4 = (112) \qquad U_{TOT}(O_4 = (112)) = 116.489$$
$$O_4 = (211) \qquad U_{TOT}(O_4 = (211)) = 100.964$$

Allora, un meccanismo deterministico ottimo, implementato secondo le strategie dominanti è dato da Tab.3.2.

	L	H
L	(222)	(122)
H	(222)	(122)

Tabella 3.11: Meccanismo deterministico ottimo implementato secondo le strategie dominanti. L'utilità attesa vale 119.916

Meccanismo deterministico, ex interim IR, implementato secondo gli equilibri di Bayes-Nash

Se supponiamo che gli agenti non conoscano quanto dichiarato dalla controparte, il risultato deve essere implementato secondo gli equilibri di Bayes-Nash. Ciascun agente, inoltre, deve ancora esere incentivato a partecipare. Bisogna dunque imporre che:

$$P(t_2 = L|o(LL) = O_1) \times u_1(L, O_1) + P(t_2 = H|o(LH) = O_3) \times u_1(L, O_3) \geq 0$$
$$P(t_2 = L|o(HL) = O_2) \times u_1(H, O_2) + P(t_2 = H|o(HH) = O_4) \times u_1(H, O_4) \geq 0$$
$$P(t_1 = L|o(LL) = O_1) \times u_2(L, O_1) + P(t_1 = H|o(HL) = O_2) \times u_2(L, O_2) \geq 0$$
$$P(t_1 = L|o(LH) = O_3) \times u_2(H, O_3) + P(t_1 = H|o(HH) = O_4) \times u_2(H, O_4) \geq 0$$

$$P(t_2 = L|o(LL) = O_1) \times u_1(L, O_1) + P(t_2 = H|o(LH) = O_3) \times u_1(L, O_3) \geq$$
$$P(t_2 = L|o(HL) = O_2) \times u_1(L, O_2) + P(t_2 = H|o(HH) = O_4) \times u_1(L, O_4)$$

$$P(t_2 = L|o(HL) = O_2) \times u_1(H, O_2) + P(t_2 = H|o(HH) = O_4) \times u_1(H, O_4) \geq$$
$$P(t_2 = L|o(LL) = O_1) \times u_1(H, O_1) + P(t_2 = H|o(LH) = O_3) \times u_1(H, O_3)$$

$$P(t_1 = L|o(LL) = O_1) \times u_2(L, O_1) + P(t_1 = H|o(HL) = O_2) \times u_2(L, O_2) \geq$$
$$P(t_1 = L|o(LH) = O_3) \times u_2(L, O_3) + P(t_1 = H|o(HH) = O_4) \times u_2(L, O_4)$$

$$P(t_1 = L|o(LH) = O_3) \times u_2(H, O_3) + P(t_1 = H|o(HH) = O_4) \times u_2(H, O_4) \geq$$
$$P(t_1 = L|o(LL) = O_1) \times u_2(H, O_1) + P(t_1 = H|o(HL) = O_2) \times u_2(H, O_2)$$

Verifichiamo se la tabella 3.10, che massimizza l'utilità attesa, è una soluzione ammissibile (se lo è, è anche soluzione ottima!). Il primo gruppo di disequazioni è banalmente soddisfatto, mentre per le disequazioni che descrivono il vincolo di compatibilità si ha:

$0.47 \times 32 + 0.53 \times 32 \geq 0.5 \times 18 + 0.53 \times 28$	$32.00 \geq 23.84$	$\surd$
$0.5 \times 74 + 0.53 \times 87 \geq 0.47 \times 68 + 0.53 \times 68$	$83.11 \geq 68.00$	$\surd$
$0.6 \times 12 + 0.73 \times 6 \geq 0.6 \times 12 + 0.63 \times 7$	$11.58 \geq 11.61$	NO!
$0.6 \times 60 + 0.63 \times 61 \geq 0.6 \times 60 + 0.73 \times 51$	$74.43 \geq 73.23$	$\surd$

Non tutte le condizioni sono soddisfatte: il secondo agente non è motivato a riportare le proprie valutazioni in modo sincero quando il suo tipo veritiero è L.
$P(t_1 = L|o(LL) = (222)) \times u_2(L, (222)) = 7.2$. Possiamo aumentarlo? No, facendo variare O_1 tra tutte le assegnazioni possibili otteniamo sempre valori inferiori a 7.2.

$O_2 = (121) \Rightarrow 4.41$	
$O_2 = (122) \Rightarrow 4.41$	
$O_2 = (221) \Rightarrow 7.20$	ma non soddisfa le condizioni BNE
$O_2 = (222) \Rightarrow 4.80$	

Otteniamo 3 soluzioni ammissibili con:

$O_2 = (121) \Rightarrow$	$U_{TOT}(121) = 112.495$
$O_2 = (122) \Rightarrow$	$U_{TOT}(122) = 119.860$
$O_2 = (222) \Rightarrow$	$U_{TOT}(222) = 107.123$

$P(t_1 = L|o(LH) = (222)) \times u_2(L, (222)) = 7.2$. Se facciamo variare O_3 per diminuire questo termine non sono più soddisfatte le condizioni BNE.
Proviamo allora con $P(t_1 = H|o(HH) = (122)) \times u_2(L, (122)) = 4.41$
Otteniamo $O_4 = (112) \Rightarrow U_{TOT}(112) = 116.489$.

Una soluzione ottima per un meccanismo deterministico implementato negli equilibri di Bayes-Nash è dato da Tab. 3.2.
Notare che è lo stesso meccanismo che si otteneva con l'implementazione nelle strategie dominanti.

	L	H
L	(222)	(122)
H	(222)	(122)

Tabella 3.12: Meccanismo deterministico ottimo implementato secondo gli equilibri di Bayes-Nash. L'utilità attesa vale 119.916

Meccanismo randomizzato, ex post IR, implementato secondo le strategie dominanti

Costruiamo ora un meccanismo randomizzato implementato secondo le strategie dominanti. Se immaginiamo di schematizzare il meccanismo prodotto con una tabella, avremo che in ogni cella non comparirà, come nei casi precedenti, un unico risultato, ma una distribuzione di probabilità su tutte le assegnazioni possibili.
Sia $P_j^1 = P(O_j = (111)),\ P_j^2 = P(O_j = (112)), \ldots, P_j^8 = P(O_j = (222))$.
Il meccanismo prodotto avrà la forma della tabella 3.13

	L	H
L	$[P_1^1 : (111)], \ldots, [P_1^8 : (222)]$	$[P_2^1 : (111)], \ldots, [P_2^8 : (222)]$
H	$[P_3^1 : (111)], \ldots, [P_3^8 : (222)]$	$[P_4^1 : (111)], \ldots, [P_4^8 : (222)]$

Tabella 3.13: Meccanismo randomizzato generico per un problema di assegnazione di valori a tre variabili

Vincolo di partecipazione "ex post IR":

$$
\begin{aligned}
&P_1^1 \times u_1(L,(111)) + P_1^2 \times u_1(L,(112)) + \cdots + P_1^8 \times u_1(L,(222)) \geq 0\\
&P_2^1 \times u_1(H,(111)) + P_2^2 \times u_1(H,(112)) + \cdots + P_2^8 \times u_1(H,(222)) \geq 0\\
&P_3^1 \times u_1(L,(111)) + P_3^2 \times u_1(L,(112)) + \cdots + P_3^8 \times u_1(L,(222)) \geq 0\\
&P_4^1 \times u_1(H,(111)) + P_4^2 \times u_1(H,(112)) + \cdots + P_4^8 \times u_1(H,(222)) \geq 0\\
&P_1^1 \times u_2(L,(111)) + P_1^2 \times u_2(L,(112)) + \cdots + P_1^8 \times u_2(L,(222)) \geq 0\\
&P_2^1 \times u_2(L,(111)) + P_2^2 \times u_2(L,(112)) + \cdots + P_2^8 \times u_2(L,(222)) \geq 0\\
&P_3^1 \times u_2(H,(111)) + P_3^2 \times u_2(H,(112)) + \cdots + P_3^8 \times u_2(H,(222)) \geq 0\\
&P_4^1 \times u_2(H,(111)) + P_4^2 \times u_2(H,(112)) + \cdots + P_4^8 \times u_2(H,(222)) \geq 0
\end{aligned}
$$

Queste condizioni sono banalmente soddisfatte in quanto, essendo $P_j^i \in [0,1]$, si sommano termini tutti positivi.

Implementazione nelle strategie dominanti:

$$P_1^1 \times u_1(L,(111)) + P_1^2 \times u_1(L,(112)) + \cdots + P_1^8 \times u_1(L,(222)) \geq$$
$$P_2^1 \times u_1(L,(111)) + P_2^2 \times u_1(L,(112)) + \cdots + P_2^8 \times u_1(L,(222))$$

$$P_3^1 \times u_1(L,(111)) + P_3^2 \times u_1(L,(112)) + \cdots + P_3^8 \times u_1(L,(222)) \geq$$
$$P_4^1 \times u_1(L,(111)) + P_4^2 \times u_1(L,(112)) + \cdots + P_4^8 \times u_1(L,(222))$$

$$P_2^1 \times u_1(H,(111)) + P_2^2 \times u_1(H,(112)) + \cdots + P_2^8 \times u_1(H,(222)) \geq$$
$$P_1^1 \times u_1(H,(111)) + P_1^2 \times u_1(H,(112)) + \cdots + P_1^8 \times u_1(H,(222))$$

$$P_4^1 \times u_1(H,(111)) + P_4^2 \times u_1(H,(112)) + \cdots + P_4^8 \times u_1(H,(222)) \geq$$
$$P_3^1 \times u_1(H,(111)) + P_3^2 \times u_1(H,(112)) + \cdots + P_3^8 \times u_1(H,(222))$$

$$P_1^1 \times u_2(L,(111)) + P_1^2 \times u_2(L,(112)) + \cdots + P_1^8 \times u_2(L,(222)) \geq$$
$$P_3^1 \times u_2(L,(111)) + P_3^2 \times u_2(L,(112)) + \cdots + P_3^8 \times u_2(L,(222))$$

$$P_2^1 \times u_2(L,(111)) + P_2^2 \times u_2(L,(112)) + \cdots + P_2^8 \times u_2(L,(222)) \geq$$
$$P_4^1 \times u_2(L,(111)) + P_4^2 \times u_2(L,(112)) + \cdots + P_4^8 \times u_2(L,(222))$$

$$P_3^1 \times u_2(H,(111)) + P_3^2 \times u_2(H,(112)) + \cdots + P_3^8 \times u_2(H,(222)) \geq$$
$$P_1^1 \times u_2(H,(111)) + P_1^2 \times u_2(H,(112)) + \cdots + P_1^8 \times u_2(H,(222))$$

$$P_4^1 \times u_2(H,(111)) + P_4^2 \times u_2(H,(112)) + \cdots + P_4^8 \times u_2(H,(222)) \geq$$
$$P_2^1 \times u_2(H,(111)) + P_2^2 \times u_2(H,(112)) + \cdots + P_2^8 \times u_2(H,(222))$$

Ossia:

$$28p_1^1 + 18p_1^2 + 36p_1^3 + 28p_1^4 + 28p_1^5 + 21p_1^6 + 37p_1^7 + 32p_1^8 \geq$$
$$28p_2^1 + 18p_2^2 + 36p_2^3 + 28p_2^4 + 28p_2^5 + 21p_2^6 + 37p_2^7 + 32p_2^8$$

$$28p_3^1 + 18p_3^2 + 36p_3^3 + 28p_3^4 + 28p_3^5 + 21p_3^6 + 37p_3^7 + 32p_3^8 \geq$$
$$28p_4^1 + 18p_4^2 + 36p_4^3 + 28p_4^4 + 28p_4^5 + 21p_4^6 + 37p_4^7 + 32p_4^8$$

$$82p_2^1 + 74p_2^2 + 74p_2^3 + 87p_2^4 + 85p_2^5 + 65p_2^6 + 69p_2^7 + 68p_2^8 \geq$$
$$82p_1^1 + 74p_1^2 + 74p_1^3 + 87p_1^4 + 85p_1^5 + 65p_1^6 + 69p_1^7 + 68p_1^8$$

$$82p_4^1 + 74p_4^2 + 74p_4^3 + 87p_4^4 + 85p_4^5 + 65p_4^6 + 69p_4^7 + 68p_4^8 \geq$$
$$82p_3^1 + 74p_3^2 + 74p_3^3 + 87p_3^4 + 85p_3^5 + 65p_3^6 + 69p_3^7 + 68p_3^8$$

$$7p_1^1 + 6p_1^2 + 7p_1^3 + 7p_1^4 + 6p_1^5 + 6p_1^6 + 12p_1^7 + 12p_1^8 \geq$$
$$7p_3^1 + 6p_3^2 + 7p_3^3 + 7p_3^4 + 6p_3^5 + 6p_3^6 + 12p_3^7 + 12p_3^8$$

$$7p_2^1 + 6p_2^2 + 7p_2^3 + 7p_2^4 + 6p_2^5 + 6p_2^6 + 12p_2^7 + 12p_2^8 \geq$$
$$7p_4^1 + 6p_4^2 + 7p_4^3 + 7p_4^4 + 6p_4^5 + 6p_4^6 + 14p_4^7 + 12p_4^8$$

$$55p_3^1 + 51p_3^2 + 53p_3^3 + 61p_3^4 + 55p_3^5 + 51p_3^6 + 53p_3^7 + 60p_3^8 \geq$$
$$55p_1^1 + 51p_1^2 + 53p_1^3 + 61p_1^4 + 55p_1^5 + 51p_1^6 + 53p_1^7 + 60p_1^8$$

$$55p_4^1 + 51p_4^4 + 53p_4^3 + 61p_4^4 + 55p_4^5 + 51p_4^6 + 53p_4^7 + 60p_4^8 \geq$$
$$55p_2^1 + 51p_2^2 + 53p_2^3 + 61p_2^4 + 55p_2^5 + 51p_2^6 + 53p_2^7 + 60p_2^8$$

Cerchiamo un meccanismo che massimizzi l'utilità attesa finale. Si ottiene la Tab. 3.14

	L	H
L	[1:(222)]	[0.83:(112)],[0.17:(221)]
H	[1:(222)]	[1:(122)]

Tabella 3.14: Meccanismo randomizzato ottimo per il problema di assegnazione di valori. L'utilità attesa vale 119.717 .

Capitolo 4

Modellazione ed implementazione del problema

Come già descritto al capitolo precedente, in questa tesi abbiamo affrontato il problema di progettare dei meccanismi di aggregazione di preferenze.
Dato un insieme di variabili, un gruppo di agenti riporta le proprie valutazioni sulle assegnazioni di valori che possono essere fatte a coppie di tali variabili, allo scopo di produrre un'assegnazione completa che soddisfi tutti gli agenti.
A differenza dei normali problemi FCSP descritti al capitolo1, i gradi di preferenza riportati dagli agenti non consistono in valutazioni numeriche, ma in tipi. Ogni tipo rappresenta un ben individuato intervallo di valori di utilità, che si suppone identico per ogni agente.
Altra caratteristica particolare consiste nel fatto che la progettazione del meccanismo di aggregazione delle preferenze deve avvenire ancora prima di conoscere le dichiarazioni dei partecipanti al sistema: si suppone che un coordinatore conosca a priori le probabilità con cui ogni agente dichiarerà ciascun tipo, e l'utilità ad esso legata. A partire da tali elementi, ci interessa progettare un meccanismo che permetta di ottenere una soluzione al problema di assegnazione, per qualsiasi combinazione di tipi. In particolare, richiediamo che tale soluzione sia una soluzione ottima, che massimizzi cioè una determinata funzione obiettivo.
Il meccanismo generato deve soddisfare alcune proprietà di non manipolabilità: poiché ogni agente conosce in anticipo i possibili risultati che ogni sua dichiarazione può produrre, proprio per il fatto che il meccanismo è costruito

ancora prima di conoscere le valutazioni, esso deve essere motivato a riportare il proprio tipo veritiero, altrimenti, agendo secondo il personale interesse, potrebbe mentire per aumentare la propria utilità, a scapito della funzione obiettivo.

In questa sezione vengono descritti in maniera approfondita gli algoritmi implementati nella tesi. La modellazione e risoluzione del problema è stata realizzata mediante ILOG CPLEX Concert Technology [3]. Il codice C++ completo si trova in Appendice.

4.1 Lettura dei dati e generazione istanza

Una prima parte del programma consiste nella generazione di un'istanza casuale del problema.
Come dati iniziali si suppone siano noti (facendoli leggere da un file di input) il numero N di agenti, il numero T di tipi (gli stessi tipi per ogni agente), il numero n di variabili, tutte con lo stesso dominio di valori ammissibili, di cui è nota la cardinalità d.
Si suppone inoltre di poter stabilire il numero di coppie di variabili su cui ciascun agente riporta le proprie valutazioni (non necessariamente le stesse coppie), fissando la densità del problema, cioé la percentuale di vincoli binari, sugli $n * (n-1)/2$ possibili.

Gli intervalli di utilità corrispondenti ai tipi vengono creati in modo casuale (ma coerentemente alla descrizione fatta al capitolo 3) come segue:
si fissa come estremo inferiore per l'utilità un valore random $Umin$ compreso tra -500 e 0, e come estremo superiore $Umax$ un valore compreso tra $Umin$ e 1000. Si suddivide poi l'intervallo $[Unim, Umax]$ in T parti uguali, ottenendo così gli intervalli relativi ai T tipi.

Vengono in seguito prodotte due matrici, le cui d^n righe corrispondo ad una enumerazione delle diverse tuple che si possono generare con le n variabili ed i d valori, mentre le $T * N$ colonne corrispondono ai tipi di ciascun agente (le prime T colonne sono i tipi del primo agente, il secondo gruppo di T colonne i tipi del secondo agente ... e così via).

In corrispondenza della cella $[i][k*T+j]$ $(i = 0, ..., d^n; k = 0...N; j = 0...T)$ ci sono le probabilità e l'utilità dell'agente k in funzione del tipo j e riguardanti la tupla i.
Queste due matrici rappresentano i dati su cui il coordinatore si basa per costruire il meccanismo.

Abbiamo già accennato al fatto (si veda pag. 41) che gli agenti non riportano le loro valutazioni sulle assegnazioni complete di valori, ma considerano in modo disgiunto alcune coppie di variabili.
Si immagina dunque che i dai forniti siano, per ogni agente, la distribuzione di probabilità sui tipi e l'utilità, relativa a ciascuna coppia valutata.
Ad esempio, per due variabili X e Y con dominio $\{0, 1\}$, abbiamo una terna <probabilità, tipo, utilità> per ogni tipo e per ogni coppia di valori delle due variabili $(a, b) \in \{0, 1\}^2$.
Nelle istanze che noi abbiamo studiato questi dati vengono generati, sempre in modo casuale, (dalla funzione *CreaIstanzaProblema*) secondo i seguenti criteri:

c_1 : Le probabilità sui tipi devono essere complementari: comprese tra 0 e 1, la loro somma deve dare 1.

c_2 : L'utilità deve appartenere ad un ben definito intervallo numerico, legato al tipo a cui si riferisce.

Queste valutazioni vengono combinate assieme per ottenere le probabilità e le utilità di ciascun tipo in relazione alle tuple complete.
La funzione *CreaIstanzaProblema* completa dunque le due matrici Utilità$[i][j]$ e Probabilità$[i][j]$ (contenenti inizialmente solo il valore nullo) nel seguente modo:

- Per ogni agente ag considera il numero di vincoli richiesto scegliendo le due variabili che compongono ciascun vincolo: (x_i, x_j);
- Analizza tutte le coppie di valori $(v_i.v_j)$ assegnabili a tali due variabili.
- Considera ogni tupla completa t: se in t viene assegnata alla coppia di variabili (x_i, x_j) proprio la coppia di valori (v_i, v_j), modifica la riga t-esima in entrambe le matrici nel seguente modo:

- per ogni tipo t dell'agente in questione somma al valore presente nella cella Utilità$[t][ag * T + t]$ l'utilità corrispondente al tipo t, divisa per il numero di vincoli del problema.
- per ogni tipo t dell'agente in questione somma al valore presente nella cella Probabilità$[t][ag * T + t]$ la probabilità corrispondente al tipo t, divisa per il numero di vincoli del problema.[1]

Il metodo con cui queste matrici vengono prodotte simula un'elaborazione delle valutazioni su singole coppie: è come se le preferenze riguardanti le assegnazioni complete di valori alle variabili fossero calcolate a partire dalle valutazioni riportate su alcune coppie di variabili.

4.2 Il meccanismo da produrre

Una volta generati i dati del problema, ossia date le matrici contenenti le utilità e le probabilità degli agenti, bisogna poter aggregare le preferenze in modo da ottenere un'assegnazione che li soddisfi.
Poiché queste preferenze rappresentano un grado di "bontà" delle assegnazioni, è ragionevole cercare, tra tutte le soluzioni possibili, quella che massimizza la soddisfazione totale.
Fissiamo dunque, come funzione obiettivo per il problema di ottimizzazione, la massimizzazione dell'utilità. Poiché il meccanismo è costruito ancora prima di conoscere le dichiarazioni degli agenti, l'utilità massimizzata sarà l'utilità attesa, pesata cioè con le probabilità di verificarsi.

4.3 Celle, tuple e vettori dei tipi

Nel progettare l'algoritmo abbiamo considerato, in molti punti, le tuple di valori ed i vettori di tipi come indici interi (di iterazione, o posizione in array) e come vettori contemporaneamente. Tale impiego è stato possibile tramite la definizione di due funzioni che permettono di trasformare valori interi in vettori e viceversa.
Durante la progettazione, infatti, non viene mai fatto uso dei valori del dominio

[1] Dividendo la probabilità e l'utilità per il numero dei vincoli, si mantengono le condizioni c1 e c2 richieste sopra.

o dei tipi, ma viene sempre considerata una loro enumerazione: $\{d_0, d_1, ..., d_k\}$ e $\{t_0, t_1, ..., t_T\}$.
Considerando solo gli indici di questi elementi, ogni vettore di tipi può essere visto come la rappresentazione in base T di un numero intero: la cella del meccanismo a cui corrisponde, mentre ogni tupla può essere vista come la rappresentazione in base d dell'indice di riga delle due matrici Utilità e Probabilità.

4.4 Non manipolabilità dei risultati

Poiché ogni agente riesce in qualsiasi momento a prendere visione degli eventuali risultati che ogni sua dichiarazione produce, egli potrebbe manipolare il risultato in proprio favore (allo scopo di massimizzare la propria utilità) riportando un tipo non veritiero. Di conseguenza, l'assegnazione prodotta dal meccanismo potrebbe non rappresentare una soluzione ottima, in quanto la non sincerità di un agente potrebbe causare una diminuzione dell'utilità dei compagni, e quindi dell'utilità totale.
In altri casi, un agente potrebbe scegliere di non partecipare al sistema, osservando come tutte le possibili soluzioni lo penalizzino. La progettazione del meccanismo fallirebbe così nell'intento di aggregare le preferenze di tutti.

Per ovviare a questi due inconvenienti, nel selezionare il risultato corrispondente a ciascun vettore di tipi[2] bisogna tener conto dei vincoli di partecipazione e di compatibilità descritti al paragrafo 3.1.
Nell'algoritmo prodotto questi vincoli sono stati implementati come modelli di programmazione lineare, sfruttando le funzioni fornite dalle librerie C++ di ILOG CPLEX Concert Technology [3] :

- Sono state scelte come variabili le probabilità con cui ciascuna tupla di valori compare in corrispondenza di un fissato vettore di tipi. Nella progettazione di un meccanismo randomizzato, queste incognite possono assumere qualsiasi valore compreso nell'intervallo $[0, 1]$, per un mecca-

[2] Ciascuna combinazione dei tipi dichiarati dall'intero gruppo di agenti.

nismo deterministico, invece, assumono solo i valori interi dell'insieme $\{0, 1\}$.

- Come funzione obiettivo è stata scelta la massimizzazione dell'utilità attesa totale, ossia la somma delle utilità dgli agenti corrispondente ad un fissato vettore di tipi, pesata con la probabilità che gli agenti dichiarino esattamente quei tipi.

- I vincoli del modello di programmazione lineare coincidono con le condizioni imposte sulle celle:

  - la somma delle variabili in una stessa cella deve dare 1;
  - l'utilità attesa in corrispondenza di ogni vettore di tipi deve essere non negativa per tutti gli agenti;
  - in ogni cella devono essere messe quelle assegnazioni che, valutate in base al tipo a cui si riferiscono, massimizzano l'utilità ottenibile con quello stesso tipo; in altre parole: non vi deve essere un risultato in un' altra cella che restituisca un utilità maggiore, se valutato con lo stesso tipo.

- Per la risoluzione abbiamo adottato l'algoritmo del Simplesso Primale, sempre impiegando le funzioni presenti in CPLEX.

4.5 Il main

Il programma principale esegue la lettura dei dati in input, gestendone gli eventuali errori. Genera un'istanza casuale del problema, risolvendolo in modo diverso, a seconda del criterio di non manipolabilità scelto, tramite una funzione switch.
Il primo parametro da passare nella linea di comando è il nome del file contenente i dati del problema: un file di testo contenente il numero di agenti, il numero di tipi, il numero di variabili, la cardinalità del dominio e la densità del problema.
Come secondo parametro esso riceve un numero intero compreso tra 0 e 8, specificante quale tipo di problema si intende risolvere:

0: Genera un unica istanza del problema e lo risolve più volte, variando il tipo di vincoli IR ed IC da soddisfare nel produrre un meccanismo ottimo. Salva su un file i diversi modelli di programmazione lineare prodotti ed il valore ottimo ottenuto per la funzione obiettivo.

1: Genera e risolve 100 iterazioni casuali di un meccanismo randomizzato, soddisfacente il vincolo "Ex Interim IR", salvando il tempo medio impiegato per trovare una soluzione ottima ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

2: Genera e risolve 100 iterazioni casuali di un meccanismo randomizzato, soddisfacente il vincolo "Ex Post IR", salvando il tempo medio impiegato per trovare una soluzione ottima ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

3: Genera e risolve 100 iterazioni casuali di un meccanismo randomizzato, soddisfacente il vincolo "Ex Interim IR" ed implementato secondo le strategie dominanti, salvando il tempo medio impiegato per trovare una soluzione ottima ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

4: Genera e risolve 100 iterazioni casuali di un meccanismo randomizzato, soddisfacente il vincolo "Ex Post IR"ed implementato secondo gli equilibri di Bayes-Nash, salvando il tempo medio impiegato per trovare una soluzione ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

5: Genera e risolve 100 iterazioni casuali di un meccanismo deterministico, soddisfacente il vincolo "Ex Interim IR", salvando il tempo medio impiegato per trovare una soluzione ottima ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

6: Genera e risolve 100 iterazioni casuali di un meccanismo deterministico, soddisfacente il vincolo "Ex Post IR", salvando il tempo medio impiegato per trovare una soluzione ottima ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

7: Genera e risolve 100 iterazioni casuali di un meccanismo deterministico, soddisfacente il vincolo "Ex Interim IR" ed implementato secondo le

strategie dominanti, salvando il tempo medio impiegato per trovare una soluzione ottima ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

8: Genera e risolve 100 iterazioni casuali di un meccanismo deterministico, soddisfacente il vincolo "Ex Post IR"ed implementato secondo gli equilibri di Bayes-Nash, salvando il tempo medio impiegato per trovare una soluzione ed il numero di soluzioni ottime trovate in queste 100 iterazioni.

Capitolo 5

Risultati sperimentali

Vediamo ora quali risultati sperimentali abbiamo ottenuto risolvendo alcuni problemi di assegnazione della forma finora descritta.

Un primo metodo di analisi consiste nel confrontare i meccanismi prodotti eseguendo l'algoritmo su un fissato problema e facendo variare le condizioni di partecipazione e compatibilità. Per ogni diversa esecuzione salviamo in un file il modello di programmazione lineare prodotto e valutiamo il valore assunto dalla funzione obiettivo in un'eventuale soluzione ottima.

Siano fissati i seguenti parametri:

- 2 agenti
- 3 tipi per ogni agente
- 3 variabili
- 2 valori nel dominio delle variabili
- una densità del problema del 100% (quindi valutazioni su 3 coppie)

Eseguendo il programma con i dati appena assegnati, la funzione generante l'istanza del problema produce le tabelle 5.1 e 5.2, contenenti l'utilità e la probabilità di ciascun agente, in funzione della tupla e del tipo a cui si riferiscono: nelle righe si legge l'assegnazione di valori alle variabili valutata, nelle colonne si legge l'agente ed il tipo corrispondente a tale valutazione. Nella cella

$[i][k * T + j]$ ci sono allora la probabilità e l'utilità dell'agente k in funzione del tipo j e riguardanti la tupla i. Il metodo con cui queste matrici vengono prodotte simula un'elaborazione delle valutazioni su singole coppie: è come se le preferenze riguardanti le assegnazioni complete di valori alle variabili fossero calcolate a partire dalle valutazioni riportate su alcune coppie di variabili.

	A_1, t_1	$A_1, t_2,$	A_1, t_3	A_2, t_1	$A_2, t_2,$	A_2, t_3
(0,0,0)	-9.267	206.369	319.886	48.326	163.856	357.971
(0,0,1)	55.74	211.727	282.082	80.575	178.412	342.258
(0,1,0)	22.346	172.503	346.827	24.645	144.664	297.575
(0,1,1)	46.326	195.195	316.198	26.818	156.072	336.524
(1,0,0)	22.239	194.773	276.324	0.317	204.57	318.366
(1,0,1)	61.311	194.123	294.06	35.387	186.468	317.35
(1,1,0)	0.508	152.92	322.572	6.922	177.408	280.027
(1,1,1)	-1.447	169.604	347.483	11.916	156.158	333.673

Tabella 5.1: Matrice rappresentante le valutazioni di 2 agenti per un sistema vincolato con 3 variabili.

	A_1, t_1	$A_1, t_2,$	A_1, t_3	A_2, t_1	$A_2, t_2,$	A_2, t_3
(0,0,0)	0.812	0.04	0.148	0.357	0.352	0.292
(0,0,1)	0.48	0.363	0.157	0.616	0.149	0.235
(0,1,0)	0.459	0.246	0.295	0.4	0.391	0.211
(0,1,1)	0.561	0.335	0.104	0.527	0.261	0.212
(1,0,0)	0.766	0.014	0.219	0.396	0.416	0.187
(1,0,1)	0.594	0.251	0.155	0.815	0.149	0.035
(1,1,0)	0.417	0.159	0.424	0.228	0.454	0.318
(1,1,1)	0.679	0.162	0.16	0.515	0.26	0.224

Tabella 5.2: Matrice delle probabilità con cui ogni agente riporta ciascun tipo, in funzione alla tupla cui si riferisce.

Supponendo, dunque, che le due matrici appena descritte rappresentino le vautazioni dei due agenti, analizziamo i vari meccanismi di aggregazione di preferenze ottenibili, confrontandone l'ottimalità.
Esaminiamo, come primo caso, il problema di progettare un meccanismo deterministico. Negli esempi che seguono non sia permessa, dunque, la rando-

mizzazione.
Richiediamo che il meccanismo prodotto soddisfi il vincolo di partecipazione nella forma "Ex post IR". Ogni agente, quindi, conosce il tipo riportato dalla controparte e può, in ogni istante, valutare l'utilità che gli perviene da ogni sua dichiarazione. Per essere incentivato a partecipare ogni risultato in tabella deve fornirgli un'utilità positiva. Il meccanismo ottimo generato per questa istanza specifica è quello di Tab. 5.3, che fornisce un'utilità attesa di $607,856$: le due dimensioni rappresentano i due agenti e le tre entrate i tipi. Nella cella $[i][j]$ si legge l'assegnazione di valori alle variabili ottenibile dalla selezione del tipo t_i da parte dell'agente A_1 e del tipo t_j da parte dell'agente A_2: (d_1, d_2, d_3) significa che alla variabile V_1 viene assegnato l'elemento d_1-esimo del dominio, a V_2 l'elemento d_2-esimo, e così via.
Per il modello di programmazione lineare prodotto dal risolutore rimandiamo all'Appendice B.

	A_1, t_1	$A_1, t_2,$	A_1, t_3
A_2, t_1	$(1,0,1)$	$(0,0,1)$	$(0,1,0)$
$A_2, t_2,$	$(1,0,0)$	$(0,1,1)$	$(1,1,0)$
A_2, t_3	$(0,0,1)$	$(0,0,1)$	$(1,1,0)$

Tabella 5.3: Meccanismo deterministico soddisfacente il vincolo "ex post IR", per il problema in questione. L'utilità attesa vale $607,856$.

Vediamo come questo valore di utilità possa essere migliorato: se noi supponiamo che gli agenti non conoscano l'uno il tipo dell'altro, ossia che venga imposto il vincolo di partecipazione nella forma "Ex interim IR", essi non possono verificare in modo immediato l'utilità che otterrebbero dichiarando un tipo piuttosto di un altro. Possono solo stimare l'utilità attesa, al variare delle possibili scelte fatte dalla controparte. In questo modo possono essere inserite nel meccanismo anche quelle assegnazioni di valori alle variabili che produrrebbero un'utilità negativa, ma che nel complesso forniscono un incentivo a partecipare, producendo un'utilità attesa positiva.
Il meccanismo prodotto, dato in Tab. 5.4, migliora così l'utilità attesa portandola al valore di $647,244$.
In entrambi i meccanismi appena considerati, però, gli agenti non sono incentivati a riportare il proprio tipo veritiero, non avendo imposto alcun vincolo di non manipolabilità.

	A_1, t_1	$A_1, t_2,$	A_1, t_3
A_2, t_1	$(1, 0, 1)$	$(0, 0, 1)$	$(0, 1, 0)$
$A_2, t_2,$	$(1, 0, 0)$	$(0, 1, 1)$	$(1, 1, 0)$
A_2, t_3	$(0, 0, 0)$	$(0, 0, 1)$	$(1, 1, 0)$

Tabella 5.4: Meccanismo deterministico soddisfacente il vincolo "ex interim IR", per il problema in questione. L'utilità attesa vale $647,244$.

A seconda di quanto si suppone che ciascun agente conosca riguardo la controparte, i meccanismi non manipolabili prodotti sono dati dalle tabelle 5.5 e 5.6.

	A_1, t_1	$A_1, t_2,$	A_1, t_3
A_2, t_1	$(0, 0, 1)$	$(0, 0, 1)$	$(0, 1, 0)$
$A_2, t_2,$	$(1, 0, 0)$	$(0, 0, 1)$	$(1, 1, 0)$
A_2, t_3	$(0, 0, 1)$	$(0, 0, 1)$	$(0, 1, 0)$

Tabella 5.5: Meccanismo deterministico soddisfacente il vincolo "ex post IR" ed implementato secondo le strategie dominanti. L'utilità attesa vale $520,334$.

	A_1, t_1	$A_1, t_2,$	A_1, t_3
A_2, t_1	$(0, 1, 0)$	$(0, 0, 0)$	$(0, 1, 0)$
$A_2, t_2,$	$(0, 1, 0)$	$(1, 1, 0)$	$(1, 1, 0)$
A_2, t_3	$(0, 1, 0)$	$(1, 1, 0)$	$(1, 1, 0)$

Tabella 5.6: Meccanismo deterministico soddisfacente il vincolo "ex interim IR" ed implementato secondo gli equilibri di Bayes-Nash. L'utilità attesa vale $391,956$.

Si nota facilmente come il vincolo di compatibilità provochi una diminuzione dell'utilità attesa: questo vincolo, infatti, comporta una restrizione dell'insieme delle possibili soluzioni inseribili in ogni cella del meccanismo, eliminando quelle assegnazioni di valori alle variabili che spingerebbero un agente a mentire, ma che aumenterebbero l'utilità attesa. A questo inconveniente si può in parte rimediare permettendo la randomizzazione dei risultati. Un meccanismo randomizzato, infatti, non assegna ad ogni vettore di tipi un determinato risultato, che nel nostro caso consiste in un'assegnazione di valori alle variabili del sistema, ma determina una distribuzione di probabilità su tutti i risultati

possibili. Quindi aumenta l'incertezza di ogni agente nel valutare i possibili risultati prodotti dalla selezione di un tipo, permettendo, cosí, di considerare anche alcune soluzioni non soddisfacenti i due vincoli IR ed IC. Il meccanismo randomizzato ottimo, implementato secondo le strategie dominanti, è specificato in tabella 5.7. L'utilità attesa vale $570,292$.

	A_1, t_1	$A_1, t_2,$	A_1, t_3
A_2, t_1	$[1 : (0,0,1)]$	$[1 : (0,0,1)]$	$[1 : (0,1,0)]$
$A_2, t_2,$	$[1 : (1,0,0)]$	$[0.25 : (0,0,0)]$ $[0.38 : (0,0,1)]$ $[0.15 : (1,0,0)]$ $[0.22 : (1,1,0)]$	$[1 : (1,1,0)]$
A_2, t_3	$[1 : (0,0,1)]$	$[1 : (0,0,1)]$	$[0.23 : (1,0,0)]$ $[0.77 : (1,1,0)]$

Tabella 5.7: Meccanismo randomizzato soddisfacente il vincolo "ex post IR" ed implementatio secondo le strategie dominanti.

Con un semplice ragionamento si può dedurre come questo comportamento valga in generale, non solo per i casi appena descritti: sia fissata un'istanza del problema di generare un meccanismo di aggregazione di preferenze. Si può immaginare una rappresentazione insiemistica dei diversi tipi di meccanismo ottenibili, come raffigurato in Fig.5.1. L'incertezza dei risultati estende l'insieme dei meccanismi ammessi, mentre il vincolo di compatibilità lo restringe. Di conseguenza l'utilità attesa risulta tanto maggiore quanto minori sono le restrizioni imposte durante la progettazione del meccanismo.

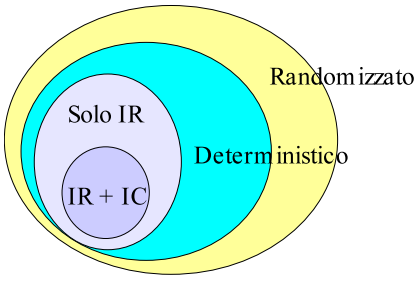


Figura 5.1: Rappresentazione insiemistica della ralazione che intercorre tra i vari tipi di meccanismo.

5.1 Analisi dei tempi di esecuzione.

Confrontiamo ora il tempo medio, su 100 iterazioni, impiegato per generare un meccanismo ottimo, variando i parametri in input e considerando i diversi modelli implementativi visti. Per ogni diversa istanza del problema consideriamo sia l'implementazione nelle strategie dominanti che l'implementazione negli equilibri di Bayes-Nash. Consideriamo inoltre sia il problema di generare un meccanismo deterministico, sia il problema di generare un meccanismo randomizzato. Per calcolare questi tempi medi, abbiamo eseguito più volte il nostro programma, salvando ogni volta il tempo impiegato per generare un meccanismo ottimo, fino ad ottenere 100 meccanismi per ogni istanza del problema.
Confrontiamo i risultati ottenuti al variare del numero di agenti, di tipi, delle variabili e della cardinalità del dominio.

Agenti:
Siano fissati i seguenti parametri:

- 3 tipi per ogni agente;
- 3 variabili;
- 2 elementi nel dominio delle variabili;
- la densità del problema sia del 100%;

La tabella 5.8 mostra il tempo medio (in 10^{-6} secondi) richiesto per ottenere un meccanismo di aggregazione delle preferenze ottimo, al variare del numero di agenti partecipanti al sistema da 2 a 5, per un meccanismo deterministico (D) o randomizzato (R), con vincolo di partecipazione "Ex post IR" (P) od "Ex interim IR" (I), implementato secondo le strategie dominanti (DS) o secondo gli equilibri di Bayes-Nash (BNE).
Soffermiamo in particolare la nostra attenzione sui meccanismi in cui sono stati imposti i vincoli di compatibilità. Confrontando il tempo medio richiesto per produrre una soluzione ottima si ottiene il grafico 5.2.
Si può facilmente constatare come il numero di agenti partecipanti al sistema influisca sul tempo necessario per ottenere un meccanismo di aggregazione, in particolar modo per la progettazione di un meccanismo che soddisfi il vincolo

di compatilbilità secondo gli equilibri di Bayes Nash.

#Ag	R/P	R/I	R/P/BNE	R/I/DS	D/P	D/I	D/P/BNE	D/I/DS
2	0.67	0.50	0.32	0.51	0.35	1.50	5.00	1.52
3	0.25	0.75	4.25	2.75	1.25	0.50	11.90	9.50
4	1.00	1.01	208.60	48.80	2.00	1.60	2644.80	426.40
5	2.50	4.00	7094.00	1517.00	6.50	2.50	32468.00	21445.00

Tabella 5.8: Il tempo medio (in 10^{-6} secondi) richiesto per risolvere 100 istanze casuali del problema di progettare un meccanismo di aggregazione delle preferenze, al variare del numero di agenti partecipanti al sistema.

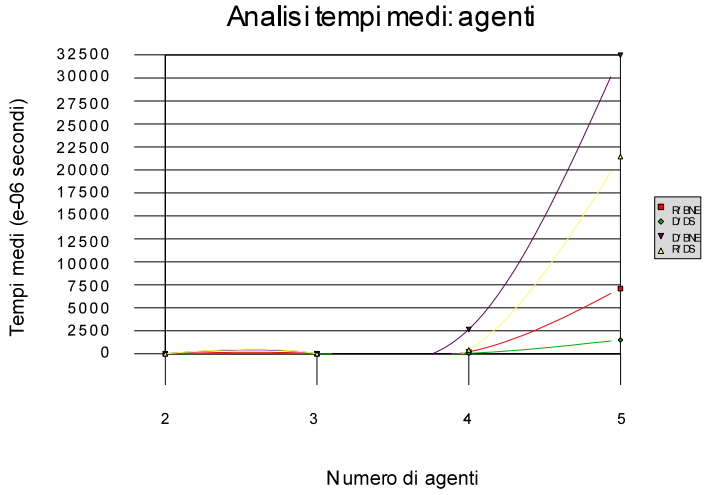


Figura 5.2: Grafico dei tempi medi per i meccanismi non manipolabili, al variare del numero degli agenti.

Tipi:

Eseguiamo lo stesso esperimento facendo variare il numero di tipi per ogni agente. Siano fissati i seguenti parametri:

- 3 agenti;
- 3 variabili;
- 2 elementi nel dominio delle variabili;

- la densità del problema sia del 100%;

La tabella 5.9 mostra il tempo medio impiegato per trovare un meccanismo ottimo, al variare del numero di tipi per ogni agente da 2 a 5.

#Tipi	R/P	R/I	R/P/BNE	R/I/DS	D/P	D/I	D/P/BNE	D/I/DS
2	0.20	0.40	0.59	1.08	0.82	0.18	3.16	2.01
3	0.25	0.75	4.25	2.75	1.25	0.50	11.90	9.50
4	1.15	1.53	12.31	6.73	2.07	0.61	164.91	18.88
5	3.02	3.13	982.00	294.00	4.00	1.00	10596.00	54.00

Tabella 5.9: Il tempo medio (in 10^{-6} secondi) richiesto per risolvere 100 istanze casuali del problema di progettare un meccanismo di aggregazione delle preferenze, al variare del numero dei tipi degli agenti, da 2 a 5.

Riassumiamo nel grafico 5.3 i risultati riguardanti i meccanismi non manipolabili.

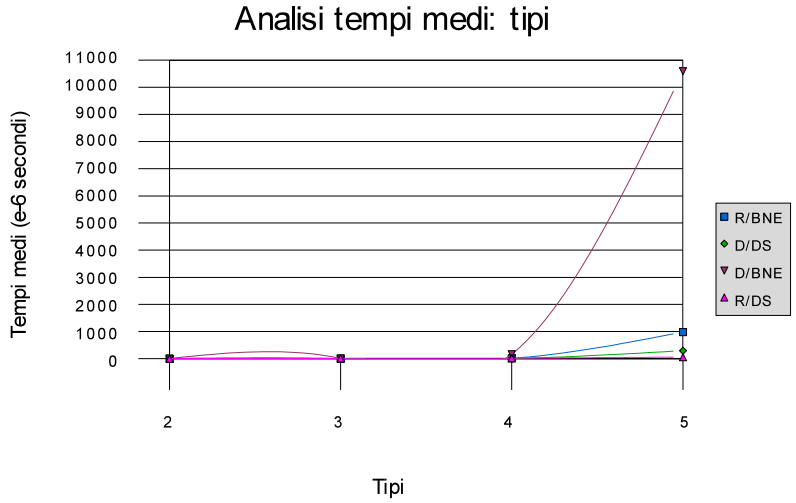


Figura 5.3: Grafico dei tempi medi per i meccanismi non manipolabili, al variare del numero di tipi per ogni agente.

Si può osservare come il numero di tipi degli agenti non influisca in maniera significativa sul tempo di esecuzione, tranne per il caso in cui si progetta un meccanismo randomizzato implementato negli equilibri di Bayes-Nash.

Cardinalità del dominio:
Fissiamo i seguenti parametri:

- 2 agenti;
- 2 tipi;
- 3 variabili;
- la densità del problema sia del 100%;

ed analizziamo come varia il tempo medio richiesto per ottenere una soluzione ottima, al variare della cardinalità dell'insieme di valori assegnabili alle variabili, da 2 a 5. La tabella 5.10 ed il grafico 5.4 mostrano come la cardinalità del dominio non sia un fattore particolarmente influenzante il tempo richiesto per ottenere un meccanismo ottimo.

#Var	R/P	R/I	R/P/BNE	R/I/DS	D/P	D/I	D/P/BNE	D/I/DS
2	0.6	0.1	0.4	0.6	0.15	0.2	1.2	1.4
3	0.01	0.01	0.5	0.25	1.5	1.33	2	1
4	0.67	1	1	1.5	1.67	1.5	3.36	4.5
5	1	0.5	0.67	1.67	2.17	1.17	7.3	5.82

Tabella 5.10: Il tempo medio (in 10^{-6} secondi) richiesto per risolvere 100 istanze casuali del problema di progettare un meccanismo di aggregazione delle preferenze, al variare della cardinalità del dominio.

Variabili:
Facciamo ora variare la dimensione del sistema vincolato e valutiamo come il numero di variabili del problema con vincoli influisca sul tempo di esecuzione. Essendo interessati in modo particolare al legame intercorrente fra il numero di variabili del sistema e la generazione dei meccanismi, approfondiamo in modo particolare quest'analisi. Supponiamo siano fissati i seguenti parametri:

- 2 tipi;
- 2 elementi nel dominio delle variabili;
- la densità del problema sia del 100%.

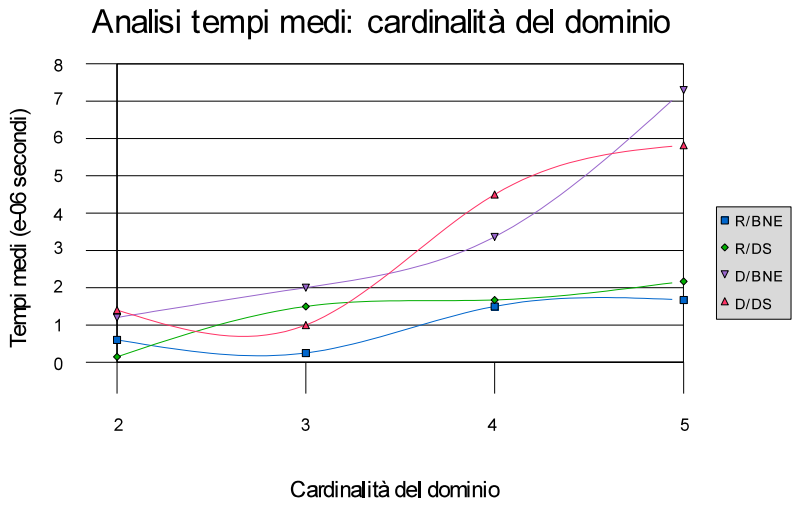


Figura 5.4: Grafico dei tempi medi per i meccanismi non manipolabili, al variare della cardinalità dell'insieme di valori ammissibili.

Consideriamo un numero di agenti variabile da 2 a 4 e valutiamo il tempo medio richiesto per ottenere un meccanismo ottimo, al variare del numero delle variabili del sistema vincolato da 3 a 10. Le tabelle 5.11, 5.12, 5.13 sintetizzano i risultati ottenuti per le istanze con un numero di agenti fissato rispettivamente a 2, 3 e 4.

#Var	R/P	R/I	R/P/BNE	R/I/DS	D/P	D/I	D/P/BNE	D/I/DS
3	0.1	0.4	0.4	0.8	0.2	0.6	1	0.8
4	0.25	0.45	0.5	0.75	1	0.5	2.75	2.5
5	1	0.5	1	1.5	0.5	0.5	2.5	1.5
6	0.1	0.5	1.5	0.5	1	0.5	2	1.9
7	1.5	1	1	1.5	2.5	0.5	8	7
8	1	1.6	1.32	1.6	2.3	1.6	15	13
9	1.5	2.5	3	2.5	6.5	1	49	30.5
10	4	4.2	7.5	7	11.5	3.2	49.8	40.7

Tabella 5.11: Il tempo medio (in 10^{-6} secondi) richiesto per risolvere 100 istanze casuali del problema di progettare un meccanismo di aggregazione delle preferenze, al variare del numero delle variabili del sistema vincolato per le istanze con 2 agenti.

Si osservi, con l'aiuto del grafici 5.1, 5.1 e 5.1, come il numero delle varia-

#Var	R/P	R/I	R/P/BNE	R/I/DS	D/P	D/I	D/P/BNE	D/I/DS
3	0.44	0.46	0.81	0.59	0.40	0.32	5.06	1.96
4	0.41	0.44	0.92	0.71	0.31	0.39	22.96	3.49
5	0.5	0.6	0.9	0.3	0.6	0.3	47.40	6.6
6	0.6	1.2	1.3	1.6	0.5	0.7	199.9	9.8
7	1.6	0.6	0.3	2.8	3.8	1.00	834	30
8	1.6	1.8	4.6	4.2	1.8	1.4	1656	45
9	4	5	11	10	3	3	7112	155
10	9.7	9.2	25	21	7	7	82064	227

Tabella 5.12: Il tempo medio (in 10^{-6} secondi) richiesto per risolvere 100 istanze casuali del problema di progettare un meccanismo di aggregazione delle preferenze, al variare del numero delle variabili del sistema vincolato per le istanze con 3 agenti.

#Var	R/P	R/I	R/P/BNE	R/I/DS	D/P	D/I	D/P/BNE	D/I/DS
3	0.4	0.3	1.5	0.7	0.3	0.2	10.3	4.6
4	0.4	0.7	3.1	1.3	0.5	0.2	269.4	20
5	0.4	0.8	3.6	2.2	0.9	0.5	971	41.6
6	0.8	0.8	5.7	4.1	2.6	0.8	13361.3	72.1
7	2.2	1.6	9.6	8.2	9.4	1.6	551543.78	251.6
8	3.8	3.8	19	18	26	0.3	13237789.5	312
9	9	8	41	30	48	7	718171982, 06	450
10	24	23	83	78	63	15	–	136896

Tabella 5.13: Il tempo medio (in 10^{-6} secondi) richiesto per risolvere 100 istanze casuali del problema di progettare un meccanismo di aggregazione delle preferenze, al variare del numero delle variabili del sistema vincolato per le istanze con 4 agenti.

bili del sistema non influenzi significativamente il tempo di esecuzione, eccetto per il caso in cui sia richiesto un meccanismo deterministico implementato secondo gli equilibri di Bayes-Nash, la cui complessità, come dimostra il grafico 5.1, cresce in modo più rapido al variare degli agenti che al variare delle variabili del sistema.

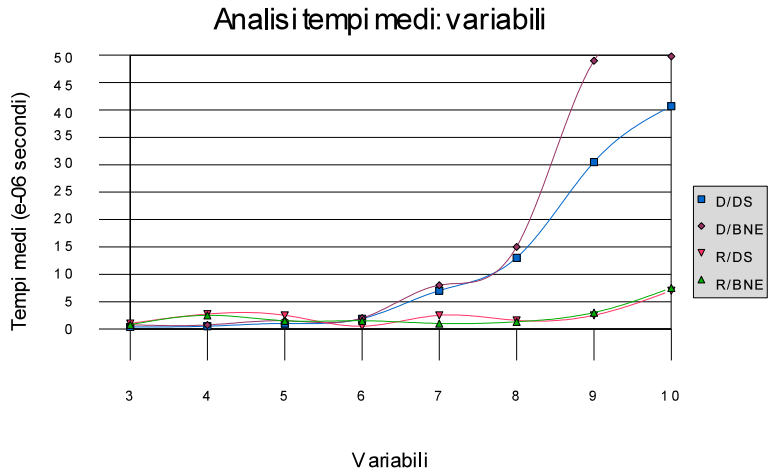


Figura 5.5: Grafico dei tempi medi per i meccanismi non manipolabili, al variare del numero di variabili del sistema, per problemi con 2 agenti. In questi grafici non viene considerato il meccanismo deterministico implementato negli equilibri di Bayes-Nash per problemi di visualizzazione. In questi grafici non viene considerato il meccanismo deterministico implementato negli equilibri di Bayes-Nash per problemi di visualizzazione.

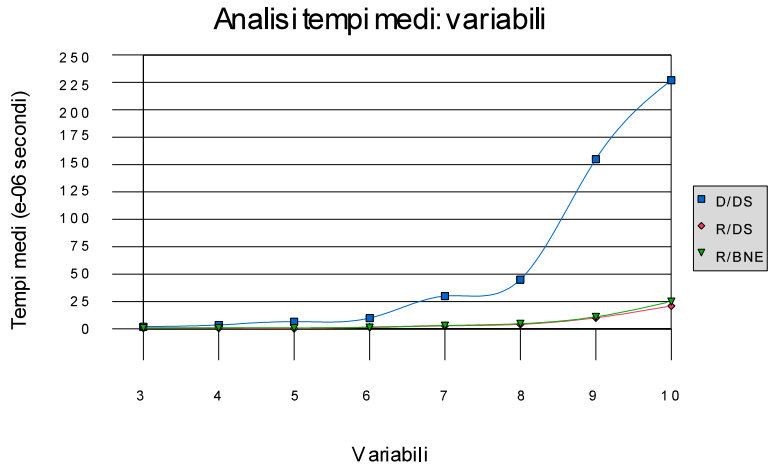


Figura 5.6: Grafico dei tempi medi per i meccanismi non manipolabili, al variare del numero di variabili del sistema, per problemi con 3 agenti.In questi grafici non viene considerato il meccanismo deterministico implementato negli equilibri di Bayes-Nash per problemi di visualizzazione.

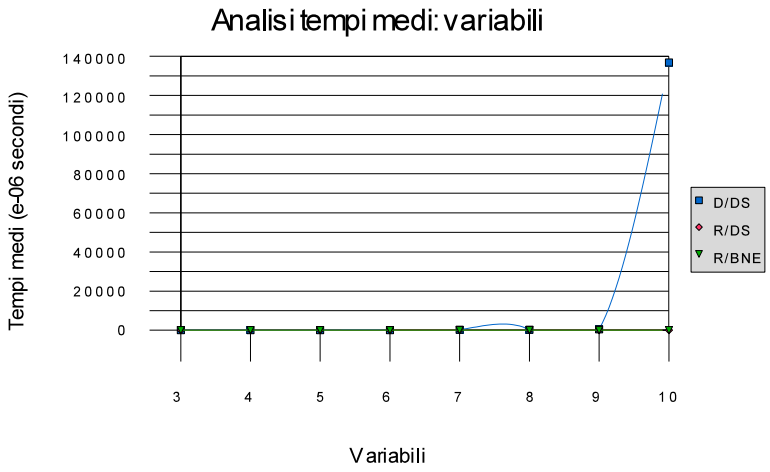


Figura 5.7: Grafico dei tempi medi per i meccanismi non manipolabili, al variare del numero di variabili del sistema, per problemi con 4 agenti.In questi grafici non viene considerato il meccanismo deterministico implementato negli equilibri di Bayes-Nash per problemi di visualizzazione.

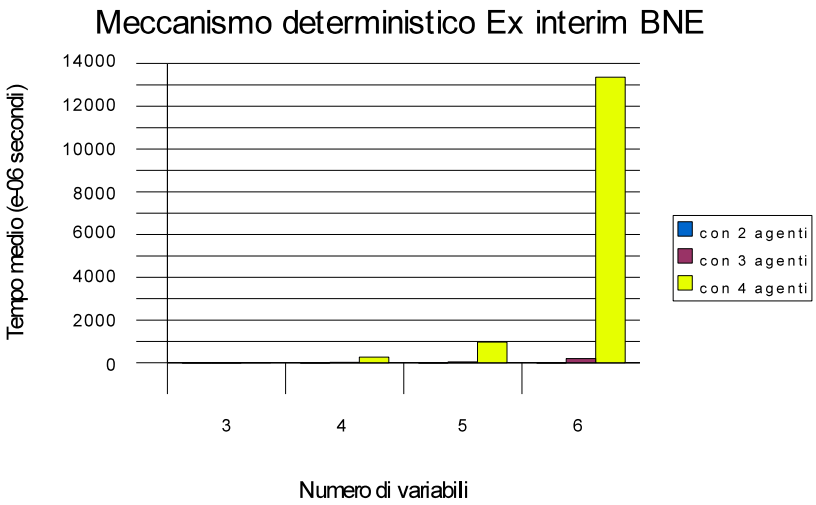


Figura 5.8: Confronto tempi medi al variare del numero di variabili del sistema e del numero di agenti, per un meccanismo BNE.In questi grafici non viene considerato il meccanismo deterministico implementato negli equilibri di Bayes-Nash per problemi di visualizzazione.

In conclusione, da questi esperimenti si può dedurre che il parametro che maggiormente influenza il tempo di esecuzione è il numero di agenti partecipanti al sistema. Il fatto che il numero di variabili non sia un parametro determinante indica che il nostro approccio, il cui scopo era quello di generare automaticamente meccanismi di aggregazione di preferenze anche per più variabili, riesce a modellizzare anche quei problemi in cui vi sono più variabili da considerare, senza far crescere la complessità.

Conclusioni

In queta tesi abbiamo considerato il problema di generare meccanismi di aggregazione di preferenze, per sistemi multi-agente con vincoli soft.
Abbiamo innanzitutto descritto i concetti principali riguardanti i problemi vincolati, osservando come il loro formalismo rigido risulti spesso inefficace nel rappresentare i problemi della vita reale.
Siamo passati così alla considerazione dei vincoli soft, una modellazione alternativa dei problemi vincolati, che permette di associare un grado di preferenza alle possibili assegnazoni di valori alle variabili.
Abbiamo, in seguito, introdotto il concetto di meccanismo non manipolabile, basandoci sugli studi compiuti da Conitzer e Sandholm in [6], cercando di adattare il modello proposto al nostro problema e di generalizzarlo alla gestione di più variabili e all'uso di preferenze su sottoinsiemi di esse.
L'osservazione di come il meccanismo studiato risulti inefficiente per quei particolari problemi in cui le valutazioni sono riportate su un gran numero di oggetti, ci ha indotti a considerare che, nonostante i risultati siano assegnazioni complete di valori a tutte le variabili del sistema, gli agenti partecipanti al meccanismo possano esprimere le loro preferenze su un piccolo insieme di tali variabili, valutate a coppie.
Infine abbiamo implementato il nostro modello servendoci di alcune funzioni della libreria C++ di ILOG CPLEX, verificandone la funzionalità tramite la generazione di istanze casuali del problema.
Gli esperimenti eseguiti hanno dimostrato come la non conoscenza, da parte degli agenti, delle dichiarazioni riportate dagli altri partecipanti, permetta di ottenere un'utilità attesa maggiore. Abbiamo riscontrato, inoltre, come la randomizzazione permetta di ridurre il tempo richiesto per ottenere un meccanismo ottimo. Infine, facendo variare i vari parametri in input (il numero di agenti, di tipi, di variabili del sistema vincolato e la dimensione del dominio),

abbiamo osservato che il tempo richiesto per generare un meccanismo ottimo non aumenta all'aumentare del numero delle variabili del sistema vincolato, concludendo così che l'approccio proposto in questa tesi soddisfa allo scopo per cui è stato ideato. Data la generalità con cui si è modellizzato il problema, questo stesso approccio può essere impiegato per risolvere dei problemi di aggregazione di preferenze con altre classi di vincoli soft. In questa tesi, infatti, abbiamo supposto che i valori di preferenza appartenessero all'intervallo $[Umin, Umax]$ e non $[0, 1]$, intervallo di preferenza del formalismo Fuzzy CSP. Inoltre ogni agente fissa in maniera autonoma il proprio operatore di combinazione, definendo così il proprio semianello di valutazioni.
Le possibili estensioni future di questo modello sono molteplici: si possono, ad esempio, generare dei formalismi specifici per risolvere particolari problemi della vita reale, dai problemi di schedulazione di corsi, a più semplici problemi di assegnazione, ampliando in tal modo il modello AMD visto in [6].

Appendice A

Implementazione del meccanismo di aggregazione di preferenze

Alcune utilità

```
#include <ilcplex/ilocplex.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<fstream.h>
#include<sstream>
#include<iostream>
#include<vector>

ILOSTLBEGIN
using namespace std;

struct Dati{
  IloInt agenti;
  IloInt tipi;
  IloInt variabili;
  IloInt dominio;
  IloInt vincoli;
  IloNum Umin;
  IloNum Umax;
};

  ofstream outputFile("Analisi", ios::app);
  ofstream dati("Dati", ios::app);
  ofstream risultati("risultati", ios::app);
```

```
//-------------------Gestione delle eccezioni---------------------------------

class FileError:public IloException{
public:
  FileError():IloException("Cannot open data file"){}};

class DataError:public IloException{
public:
  DataError():IloException("Error: the values in input for agents, types,
  variables and dimension of domain must be positive"){}};

class UtilityError:public IloException{
 public:
  UtilityError():IloException("Error: utility domain must be a interval"){}};

class DensityError:public IloException{
public:
  DensityError():IloException("Error: the value of density must be in [0,100]"){}};

//----------------------Alcune funzioni utili-------------------------------

IloNum ArrotondaDecimali(IloNum numero){
  IloNum A=numero*1000;
  IloInt B=(IloInt)A;
  IloNum C=(IloNum)B;
  if(A-C>=0.55) C++;
  C=C/1000;
  return C;}

IloNum Random(IloNum min, IloNum Max){
  IloNum n = ((IloNum)random())/RAND_MAX;
  n *= (Max-min);
  n += min;
  return ArrotondaDecimali(n);}

bool Base_n(const Dati& problema, IloInt numero, IloInt A, IloInt B,
  IloInt primo, IloInt secondo){
  IloInt X=0,Y=0;
  for(IloInt i=0; i<problema.variabili; i++){
    IloInt j=0;
    while(numero>=(IloInt)IloPower(problema.dominio,(problema.variabili-1-i))){
      numero-=(IloInt)IloPower(problema.dominio,(problema.variabili-1-i));
```

```
      j++;}
    if (j>=problema.dominio)cout<<"Errore!! dovrebbe essere un numero in base "
        <<problema.dominio<<"!!!"<<endl;
    if(i==A) X=j;
    if(i==B) Y=j;}
  if(primo==X && secondo==Y)return true;
  return false;}

void DaIntAVettoreTipi(const Dati& problema, IloInt valore, IloInt* T){
  for (IloInt i=0; i<problema.agenti; i++){
    T[problema.agenti-1-i] = valore-(IloInt)(valore/problema.tipi)*problema.tipi;
    valore/=problema.tipi;}
  return;}

IloInt DaVettoreAInt(const Dati& problema, IloInt* T){
  IloInt valore = 0;
  for (IloInt i=0; i<problema.agenti; i++){
    if(i < problema.agenti) valore*=problema.tipi;
    valore += T[i];}
  return valore;}

//------------------------------------------------------------------------

void CreaIstanzaProblema(IloEnv env, const Dati& problema,
IloNum** Utilita, IloNum**  Probabilita){
  IloNum U=((problema.Umax)-(problema.Umin))/problema.tipi;
  IloNum utilita, probabilita, prob, util;
  for(IloInt agente=1; agente<=problema.agenti; agente++){  //Per ogni agente
    IloInt ** Grafo;
    Grafo= new (IloInt *)[problema.variabili];
    for(IloInt i=0;i<problema.variabili;i++){Grafo[i]=new IloInt[problema.variabili];}
    for(IloInt i=0;i<problema.variabili;i++){
      for(IloInt j=0;j<problema.variabili;j++){
Grafo[i][j]=0;}
      Grafo[i][i]=1;}
    IloInt vincolidavalutare = problema.vincoli;
    while(vincolidavalutare>0){                          // Valuto tutti i vincoli
    rand:
      IloInt primaVariabile=random()%problema.variabili;
      IloInt secondaVariabile=random()%problema.variabili;
      if(Grafo[primaVariabile][secondaVariabile]!=0) goto rand;
      Grafo[primaVariabile][secondaVariabile]=1;
```

```
      Grafo[secondaVariabile][primaVariabile]=1;
      vincolidavalutare-=1;
      for(IloInt primoValore=0; primoValore<problema.dominio; primoValore++){
         for(IloInt secondoValore=0; secondoValore<problema.dominio;
 secondoValore++){
  IloNum somma = 0;
  for(IloInt tipo=0; tipo<problema.tipi; tipo++){    //per ogni tipo
    IloInt colonnaDaModificare = (agente-1)*problema.tipi + tipo;
    if(tipo==problema.tipi-1){
      prob = 1-somma;
      util = Random(problema.Umin + (problema.tipi-1)*U, problema.Umin +
 problema.tipi*U);}
    else {
      prob = Random(0,1-somma);
      somma += prob;
      util = Random(problema.Umin + tipo*U, problema.Umin + (tipo+1)*U-0.01);}
    probabilita = ArrotondaDecimali(prob/problema.vincoli);
    utilita = ArrotondaDecimali(util/problema.vincoli);
    for(IloInt tupla=0; tupla<(IloInt)IloPower(problema.dominio,
problema.variabili); tupla++){
      if(Base_n(problema, tupla, primaVariabile, secondaVariabile,
primoValore, secondoValore)){
IloInt rigaDaModificare = tupla;
Utilita[rigaDaModificare][colonnaDaModificare]+= utilita;
Probabilita[rigaDaModificare][colonnaDaModificare]+=probabilita;
}}}}}}}
  dati<<"Matrice utilita':"<<endl<<endl;
  for(int i =0; i<(IloInt)IloPower(problema.dominio, problema.variabili);i++){
    for(int j=0; j<problema.tipi*problema.agenti; j++){
      dati<<Utilita[i][j]<<"  ";}
    dati<<endl;}
  dati<<endl<<endl<<"Matrice probabilita':"<<endl<<endl;
  for(int i =0; i<(IloInt)IloPower(problema.dominio, problema.variabili);i++){
    for(int j=0; j<problema.tipi*problema.agenti; j++){
      dati<<Probabilita[i][j]<<"  ";}
    dati<<endl;}
}// end CreaIstanzaProblema
```

I vincoli di partecipazione e di compatibilità

```
//------------RDNExINTERIM-------------------------------------------

static void RandomExInterim(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  IloInt disequazioni = problema.agenti*problema.tipi;
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt incognite = (IloInt)IloPower(problema.dominio, problema.variabili)*
(IloInt)IloPower(problema.tipi, problema.agenti);
  for(IloInt i0 = 0; i0 < disequazioni; i0++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i1 = 0; i1 < equazioni; i1++){c.add(IloRange(env, 1.0, 1.0));}
  for(IloInt i2 = 0; i2 < incognite; i2++){
    x.add(IloNumVar(env, 0.0, 1.0));}
  for(IloInt ivar = 0; ivar < incognite; ivar++){
    IloInt cella = ivar/((IloInt)IloPower(problema.dominio,problema.variabili));
    IloInt  tupla = ivar%((IloInt)IloPower(problema.dominio,problema.variabili));
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloNum P=1,U=0;
    for(IloInt iag = 0; iag < problema.agenti; iag++){
      IloNum coef = 1;
      IloInt vincolo = iag*problema.tipi + vettoretipi[iag];
      for(IloInt j = 0; j < problema.agenti; j++){
if(j!=iag){
  coef*=Probabilita[tupla][j*problema.tipi + vettoretipi[j]];}}
      coef*=Utilita[tupla][vincolo];
      c[vincolo].setCoef(x[ivar],ArrotondaDecimali(coef));
      P*=Probabilita[tupla][vincolo];
      U+=Utilita[tupla][vincolo];}
    c[problema.agenti*problema.tipi + cella].setCoef(x[ivar],1.0);
    obj.setCoef(x[ivar],ArrotondaDecimali(P*U));}}


//--------------RDNexPOST-----------------------------------------

static void RandomExPost(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt disequazioni = problema.agenti*equazioni;
  IloInt incognite = (IloInt)IloPower(problema.dominio, problema.variabili)*
```

```
(IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  for(IloInt i = 0; i < disequazioni; i++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i = 0; i < equazioni; i++){c.add(IloRange(env, 1.0, 1.0));}
  for(IloInt i = 0; i < incognite; i++){
   x.add(IloNumVar(env, 0.0, 1.0));
    IloNum U = 0;
    IloNum P = 1;
    IloInt cella = i/(IloInt)IloPower(problema.dominio,problema.variabili);
    IloInt tupla = i%(IloInt)IloPower(problema.dominio,problema.variabili);
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloInt eq = (IloInt)IloPower(problema.tipi,problema.agenti)*problema.agenti + cella;
    for(IloInt ag = 0; ag < problema.agenti; ag++){
      IloInt diseq = cella*problema.agenti + ag;
      c[diseq].setCoef(x[i], Utilita[tupla][vettoretipi[ag] + ag*problema.tipi]);
      U += Utilita[tupla][vettoretipi[ag] + ag*problema.tipi];
      P *= Probabilita[tupla][vettoretipi[ag] + ag*problema.tipi];}
    c[eq].setCoef(x[i], 1.0);
    obj.setCoef(x[i], ArrotondaDecimali(P*U));}}


//----------------------RDNexINTERIM- BNE-------------------------------------

static void RandomExInterimBNE(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  IloInt disequazioni = problema.agenti*problema.tipi;
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt incognite = (IloInt)IloPower(problema.dominio, problema.variabili)*
(IloInt)IloPower(problema.tipi, problema.agenti);
  for(IloInt i0 = 0; i0 < disequazioni; i0++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i1 = 0; i1 < equazioni; i1++){c.add(IloRange(env, 1.0, 1.0));}
  //  for(IloInt i2 = 0; i2 < incognite; i2++){}
  for(IloInt ivar = 0; ivar < incognite; ivar++){
    x.add(IloNumVar(env, 0.0, 1.0));
    IloInt cella = ivar/((IloInt)IloPower(problema.dominio,problema.variabili));
    IloInt  tupla = ivar%((IloInt)IloPower(problema.dominio,problema.variabili));
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloNum P=1,U=0;
```

```
    for(IloInt iag = 0; iag < problema.agenti; iag++){
      IloNum coef = 1;
      IloInt vincolo = iag*problema.tipi + vettoretipi[iag];
      for(IloInt j = 0; j < problema.agenti; j++){
if(j!=iag){
  coef*=Probabilita[tupla][j*problema.tipi + vettoretipi[j]];}}
      coef*=Utilita[tupla][vincolo];
      c[vincolo].setCoef(x[ivar],ArrotondaDecimali(coef));
      P*=Probabilita[tupla][vincolo];
      U+=Utilita[tupla][vincolo];}
    c[problema.agenti*problema.tipi + cella].setCoef(x[ivar],1.0);
    obj.setCoef(x[ivar],ArrotondaDecimali(P*U));}
  for(IloInt vin=0; vin<problema.agenti*(problema.tipi-1)*problema.tipi; vin++){
    c.add(IloRange(env, 0.0, IloInfinity));
    IloInt agente= vin/((problema.tipi-1)*problema.tipi);
    IloInt tipo= (vin/(problema.tipi-1))%problema.tipi;
    for(IloInt var=0; var<incognite; var++){
      IloNum coef=1;
      IloInt cella = var/((IloInt)IloPower(problema.dominio,problema.variabili));
      IloInt  tupla = var%((IloInt)IloPower(problema.dominio,problema.variabili));
      DaIntAVettoreTipi(problema, cella, vettoretipi);
      for(IloInt ag=0; ag<problema.agenti; ag++){
IloInt y=(ag-1)*problema.tipi + vettoretipi[ag];
if(ag==agente){coef*=Utilita[tupla][y];}
else {coef*=Probabilita[tupla][y];}}
      if(tipo!=vettoretipi[agente]){coef*=(-1);}
      c[vin+equazioni+disequazioni].setCoef(x[var],ArrotondaDecimali(coef));}}}

//---------------------- RDNexPOST_DS ----------------------------------

static void RandomExPostDS(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt dn = (IloInt)IloPower(problema.dominio,problema.variabili);
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt disequazioni = problema.agenti*equazioni;
  IloInt incognite = dn*equazioni;
  IloInt vincoliIC = problema.agenti*(problema.tipi-1)*equazioni;
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  for(IloInt i = 0; i < disequazioni; i++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i = 0; i < equazioni; i++){c.add(IloRange(env, 1.0, 1.0));}
```

```
  for(IloInt i = 0; i < vincoliIC; i++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i = 0; i < incognite; i++){
    x.add(IloNumVar(env, 0.0, 1.0));
    IloNum U = 0;
    IloNum P = 1;
    IloInt cella = i/dn;
    IloInt tupla = i%dn;
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloInt eq = equazioni*problema.agenti + cella;
    for(IloInt ag = 0; ag < problema.agenti; ag++){
      IloInt diseq = cella*problema.agenti + ag;
      c[diseq].setCoef(x[i], Utilita[tupla][vettoretipi[ag] + ag*problema.tipi]);
      U += Utilita[tupla][vettoretipi[ag] + ag*problema.tipi];
      P *= Probabilita[tupla][vettoretipi[ag] + ag*problema.tipi];}
    c[eq].setCoef(x[i], 1.0);
    obj.setCoef(x[i], ArrotondaDecimali(P*U));}
  IloInt vin = equazioni + disequazioni;
  for(IloInt ag = 0; ag < problema.agenti; ag++){
    for(IloInt cella = 0; cella < equazioni; cella++){
      DaIntAVettoreTipi(problema, cella, vettoretipi);
      IloInt tipo = vettoretipi[ag];
      for(IloInt t = 0; t < problema.tipi; t++){
if(t!=tipo){
  vettoretipi[ag]=t;
  IloInt alternativa = DaVettoreAInt(problema, vettoretipi);
  for(IloInt variabile = 0; variabile < dn; variabile++){
    c[vin].setCoef(x[variabile+cella*dn],
    Utilita[variabile][ag*problema.tipi+tipo]);
    c[vin].setCoef(x[variabile+alternativa*dn],
   -Utilita[variabile][ag*problema.tipi+tipo]);}
  vin++; }}}}}
```

```
//------------DETexINTERIM-----------------------------------------

static void DetExInterim(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
```

```
  IloInt disequazioni = problema.agenti*problema.tipi;
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt incognite = (IloInt)IloPower(problema.dominio, problema.variabili)*
(IloInt)IloPower(problema.tipi, problema.agenti);
  for(IloInt i0 = 0; i0 < disequazioni; i0++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i1 = 0; i1 < equazioni; i1++){c.add(IloRange(env, 1.0, 1.0));}
  for(IloInt i2 = 0; i2 < incognite; i2++){x.add(IloIntVar(env, 0, 1,ILOINT));}
  for(IloInt ivar = 0; ivar < incognite; ivar++){
    IloInt cella = ivar/((IloInt)IloPower(problema.dominio,problema.variabili));
    IloInt  tupla = ivar%((IloInt)IloPower(problema.dominio,problema.variabili));
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloNum P=1,U=0;
    for(IloInt iag = 0; iag < problema.agenti; iag++){
      IloNum coef = 1;
      IloInt vincolo = iag*problema.tipi + vettoretipi[iag];
      for(IloInt j = 0; j < problema.agenti; j++){
if(j!=iag){
  coef*=Probabilita[tupla][j*problema.tipi + vettoretipi[j]];}}
      coef*=Utilita[tupla][vincolo];
      c[vincolo].setCoef(x[ivar],ArrotondaDecimali(coef));
      P*=Probabilita[tupla][vincolo];
      U+=Utilita[tupla][vincolo];}
    c[problema.agenti*problema.tipi + cella].setCoef(x[ivar],1.0);
    obj.setCoef(x[ivar],ArrotondaDecimali(P*U));}}


//---------------DETexPOST-----------------------------------------------

static void DetExPost(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt disequazioni = problema.agenti*equazioni;
  IloInt incognite = (IloInt)IloPower(problema.dominio, problema.variabili)*
(IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  for(IloInt i = 0; i < disequazioni; i++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i = 0; i < equazioni; i++){c.add(IloRange(env, 1.0, 1.0));}
  for(IloInt i = 0; i < incognite; i++){
   x.add(IloIntVar(env, 0, 1));
    IloNum U = 0;
```

```
    IloNum P = 1;
    IloInt cella = i/(IloInt)IloPower(problema.dominio,problema.variabili);
    IloInt tupla = i%(IloInt)IloPower(problema.dominio,problema.variabili);
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloInt eq = (IloInt)IloPower(problema.tipi,problema.agenti)*problema.agenti
     + cella;
    for(IloInt ag = 0; ag < problema.agenti; ag++){
      IloInt diseq = cella*problema.agenti + ag;
      c[diseq].setCoef(x[i], Utilita[tupla][vettoretipi[ag] + ag*problema.tipi]);
      U += Utilita[tupla][vettoretipi[ag] + ag*problema.tipi];
      P *= Probabilita[tupla][vettoretipi[ag] + ag*problema.tipi];}
    c[eq].setCoef(x[i], 1.0);
    obj.setCoef(x[i], ArrotondaDecimali(P*U));}}


//----------------------DETexINTERIM- BNE-------------------------------------

static void DetExInterimBNE(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  IloInt disequazioni = problema.agenti*problema.tipi;
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt incognite = (IloInt)IloPower(problema.dominio, problema.variabili)*
(IloInt)IloPower(problema.tipi, problema.agenti);
  for(IloInt i0 = 0; i0 < disequazioni; i0++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i1 = 0; i1 < equazioni; i1++){c.add(IloRange(env, 1.0, 1.0));}
  for(IloInt i2 = 0; i2 < incognite; i2++){x.add(IloIntVar(env, 0, 1,ILOINT));}
  for(IloInt ivar = 0; ivar < incognite; ivar++){
    IloInt cella = ivar/((IloInt)IloPower(problema.dominio,problema.variabili));
    IloInt  tupla = ivar%((IloInt)IloPower(problema.dominio,problema.variabili));
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloNum P=1,U=0;
    for(IloInt iag = 0; iag < problema.agenti; iag++){
      IloNum coef = 1;
      IloInt vincolo = iag*problema.tipi + vettoretipi[iag];
      for(IloInt j = 0; j < problema.agenti; j++){
if(j!=iag){
  coef*=Probabilita[tupla][j*problema.tipi + vettoretipi[j]];}}
      coef*=Utilita[tupla][vincolo];
      c[vincolo].setCoef(x[ivar],ArrotondaDecimali(coef));
```

```
      P*=Probabilita[tupla][vincolo];
      U+=Utilita[tupla][vincolo];}
    c[problema.agenti*problema.tipi + cella].setCoef(x[ivar],1.0);
    obj.setCoef(x[ivar],ArrotondaDecimali(P*U));}
  for(IloInt vin=0; vin<problema.agenti*(problema.tipi-1)*problema.tipi; vin++){
    c.add(IloRange(env, 0.0, IloInfinity));
    IloInt agente= vin/((problema.tipi-1)*problema.tipi);
    IloInt tipo= (vin/(problema.tipi-1))%problema.tipi;
    for(IloInt var=0; var<incognite; var++){
      IloNum coef=1;
      IloInt cella = var/((IloInt)IloPower(problema.dominio,problema.variabili));
      IloInt  tupla = var%((IloInt)IloPower(problema.dominio,problema.variabili));
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    for(IloInt ag=0; ag<problema.agenti; ag++){
      IloInt y=(ag-1)*problema.tipi + vettoretipi[ag];
      if(ag==agente){coef*=Utilita[tupla][y];}
      else {coef*=Probabilita[tupla][y];}}
    if(tipo!=vettoretipi[agente]){coef*=(-1);}
    c[vin+equazioni+disequazioni].setCoef(x[var],ArrotondaDecimali(coef));}}}

//---------------------- DETexPOST_DS ----------------------------------


static void DetExPostDS(IloEnv env, IloNumVarArray x, IloRangeArray c,
IloObjective obj, const Dati& problema, IloNum** Utilita,
IloNum**  Probabilita){
  IloInt dn = (IloInt)IloPower(problema.dominio,problema.variabili);
  IloInt equazioni = (IloInt)IloPower(problema.tipi, problema.agenti);
  IloInt disequazioni = problema.agenti*equazioni;
  IloInt incognite = dn*equazioni;
  IloInt vincoliIC = problema.agenti*(problema.tipi-1)*equazioni;
  IloInt* vettoretipi;
  vettoretipi = new(IloInt)[problema.agenti];
  for(IloInt i = 0; i < disequazioni; i++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i = 0; i < equazioni; i++){c.add(IloRange(env, 1.0, 1.0));}
  for(IloInt i = 0; i < vincoliIC; i++){c.add(IloRange(env, 0.0, IloInfinity));}
  for(IloInt i = 0; i < incognite; i++){
    x.add(IloIntVar(env, 0, 1,ILOINT));
    IloNum U = 0;
    IloNum P = 1;
    IloInt cella = i/dn;
    IloInt tupla = i%dn;
```

```
    DaIntAVettoreTipi(problema, cella, vettoretipi);
    IloInt eq = equazioni*problema.agenti + cella;
    for(IloInt ag = 0; ag < problema.agenti; ag++){
      IloInt diseq = cella*problema.agenti + ag;
      c[diseq].setCoef(x[i], Utilita[tupla][vettoretipi[ag] + ag*problema.tipi]);
      U += Utilita[tupla][vettoretipi[ag] + ag*problema.tipi];
      P *= Probabilita[tupla][vettoretipi[ag] + ag*problema.tipi];}
    c[eq].setCoef(x[i], 1.0);
    obj.setCoef(x[i], ArrotondaDecimali(P*U));}
  IloInt vin = equazioni + disequazioni;
  for(IloInt ag = 0; ag < problema.agenti; ag++){
    for(IloInt cella = 0; cella < equazioni; cella++){
      DaIntAVettoreTipi(problema, cella, vettoretipi);
      IloInt tipo = vettoretipi[ag];
      for(IloInt t = 0; t < problema.tipi; t++){
if(t!=tipo){
  vettoretipi[ag]=t;
  IloInt alternativa = DaVettoreAInt(problema, vettoretipi);
  for(IloInt variabile = 0; variabile < dn; variabile++){
    c[vin].setCoef(x[variabile+cella*dn],
    Utilita[variabile][ag*problema.tipi+tipo]);
    c[vin].setCoef(x[variabile+alternativa*dn],
  -Utilita[variabile][ag*problema.tipi+tipo]);}
  vin++; }}}}}
```

Il programma principale e l'analisi dei vincoli

```
void AnalizzaVincoli(const Dati& problema, IloNum** Utilita, IloNum**  Probabilita){
try{
  IloEnv env1;
  IloNumVarArray var1(env1);
  IloRangeArray rng1(env1);
  IloObjective obj1 = IloMaximize(env1);
  RandomExPost(env1, var1, rng1, obj1, problema, Utilita, Probabilita);
  ofstream risultati("RandomExPost.txt", ios::app);
  risultati<<"Random Ex Post:"<<endl<<endl;
  IloModel model1(env1, "RandomExPost");
  model1.add(obj1);
  model1.add(rng1);
```

```
IloCplex cplex1(model1) ;
cplex1.exportModel("RandomExPost.lp");
cplex1.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals1(env1);
if(!cplex1.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex1.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex1.getObjValue() << endl;
  cplex1.getValues(vals1, var1);
  risultati << "Values        = " << vals1 << endl<<endl;}
risultati << "_____________________________________"<<endl<<endl;
env1.end();

IloEnv env2;
IloNumVarArray var2(env2);
IloRangeArray rng2(env2);
IloObjective obj2 = IloMaximize(env2);
RandomExInterim(env2, var2, rng2, obj2, problema, Utilita, Probabilita);
ofstream risultati("RandomExinterim.txt", ios::app);
risultati<<"Random Ex interim:"<<endl<<endl;
IloModel model2(env2, "RandomExInterim");
model2.add(obj2);
model2.add(rng2);
IloCplex cplex2(model2) ;
cplex2.exportModel("RandomExInterim.lp");
cplex2.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals2(env2);
if(!cplex2.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex2.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex2.getObjValue() << endl;
  cplex2.getValues(vals2, var2);
  risultati << "Values        = " << vals2 << endl<<endl;}
risultati <<"_____________________________________"<<endl<<endl;
env2.end();

IloEnv env3;
IloNumVarArray var3(env3);
IloRangeArray rng3(env3);
IloObjective obj3 = IloMaximize(env3);
RandomExPostDS(env3, var3, rng3, obj3, problema, Utilita, Probabilita);
ofstream risultati("RandomExPostDS.txt", ios::app);
```

```
risultati<<"Random Ex Post DS:"<<endl<<endl;
IloModel model3(env3, "RandomExPostDS");
model3.add(obj3);
model3.add(rng3);
IloCplex cplex3(model3) ;
cplex3.exportModel("RandomExPostDS.lp");
cplex3.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals3(env3);
if(!cplex3.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex3.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex3.getObjValue() << endl;
  cplex3.getValues(vals3, var3);
  risultati << "Values        = " << vals3 << endl<<endl;}
risultati <<"_____________________________________"<<endl<<endl;
env3.end();

IloEnv env4;
IloNumVarArray var4(env4);
IloRangeArray rng4(env4);
IloObjective obj4 = IloMaximize(env4);
RandomExInterimBNE(env4, var4, rng4, obj4, problema, Utilita, Probabilita);
ofstream risultati("RandomExInterimBNE.txt", ios::app);
risultati<<"Random Ex Interim BNE:"<<endl<<endl;
IloModel model4(env4, "RandomExInterimBNE");
model4.add(obj4);
model4.add(rng4);
IloCplex cplex4(model4) ;
cplex4.exportModel("RandomExInterimBNE.lp");
cplex4.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals4(env4);
if(!cplex4.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex4.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex4.getObjValue() << endl;
  cplex4.getValues(vals4, var4);
  risultati << "Values        = " << vals4 << endl<<endl;}
risultati <<"_____________________________________"<<endl<<endl;
env4.end();

IloEnv env5;
IloNumVarArray var5(env5);
```

```
IloRangeArray rng5(env5);
IloObjective obj5 = IloMaximize(env5);
DetExPost(env5, var5, rng5, obj5, problema, Utilita, Probabilita);
ofstream risultati("DetExPost.txt", ios::app);
risultati<<"Det Ex Post:"<<endl<<endl;
IloModel model5(env5, "DetExPost");
model5.add(obj5);
model5.add(rng5);
IloCplex cplex5(model5) ;
cplex5.exportModel("DetExPost.lp");
cplex5.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals5(env5);
if(!cplex5.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex5.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex5.getObjValue() << endl;
  cplex5.getValues(vals5, var5);
  risultati << "Values        = " << vals5 << endl<<endl;}
risultati <<"_____________________________________"<<endl<<endl;
env5.end();

IloEnv env6;
IloNumVarArray var6(env6);
IloRangeArray rng6(env6);
IloObjective obj6 = IloMaximize(env6);
DetExInterim(env6, var6, rng6, obj6, problema, Utilita, Probabilita);
ofstream risultati("DetExinterim.txt", ios::app);
risultati<<"Det Ex interim:"<<endl<<endl;
IloModel model6(env6, "DetExInterim");
model6.add(obj6);
model6.add(rng6);
IloCplex cplex6(model6) ;
cplex6.exportModel("DetExInterim.lp");
cplex6.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals6(env6);
if(!cplex6.solve()){env6.error()<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex6.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex6.getObjValue() << endl;
  cplex6.getValues(vals6, var6);
  risultati << "Values        = " << vals6 << endl<<endl;}
risultati <<"_____________________________________"<<endl<<endl;
```

```
env6.end();

IloEnv env7;
IloNumVarArray var7(env7);
IloRangeArray rng7(env7);
IloObjective obj7 = IloMaximize(env7);
DetExPostDS(env7, var7, rng7, obj7, problema, Utilita, Probabilita);
ofstream risultati("DetExPostDS.txt", ios::app);
risultati<<"Det Ex Post DS:"<<endl<<endl;
IloModel model7(env7, "DetExPostDS");
model7.add(obj7);
model7.add(rng7);
IloCplex cplex7(model7) ;
cplex7.exportModel("DetExPostDS.lp");
cplex7.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals7(env7);
if(!cplex7.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex7.getStatus()<<endl;
  risultati<< "Solution value  = " << cplex7.getObjValue() << endl;
  cplex7.getValues(vals7, var7);
  risultati << "Values        = " << vals7 << endl<<endl;}
risultati <<"_____________________________________"<<endl<<endl;
env7.end();

IloEnv env8;
IloNumVarArray var8(env8);
IloRangeArray rng8(env8);
IloObjective obj8 = IloMaximize(env8);
DetExInterimBNE(env8, var8, rng8, obj8, problema, Utilita, Probabilita);
ofstream risultati("DetExInterimBNE.txt", ios::app);
risultati<<"Det Ex Interim BNE:"<<endl<<endl;
IloModel model8(env8, "DetExInterimBNE");
model8.add(obj8);
model8.add(rng8);
IloCplex cplex8(model8) ;
cplex8.exportModel("DetExInterimBNE.lp");
cplex8.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
IloNumArray vals8(env8);
if(!cplex8.solve()){risultati<<"Failed to optimize "<<endl;}
else{
  risultati<<"Solution status ="<<cplex8.getStatus()<<endl;
```

```
    risultati<< "Solution value  = " << cplex8.getObjValue() << endl;
    cplex8.getValues(vals8, var8);
    risultati << "Values        = " << vals8 << endl<<endl;}
  risultati <<"_____________________________________"<<endl<<endl;
  env8.end();}
catch (IloException& ex){}
catch (...){cerr<<"Unknown exception caught."<<endl;}
}

//----------------Main-----------------------------------------------------

int main(IloInt argc, char * argv[]){
  IloEnv env;
  try {
    Dati problema;
    ifstream inputFile(argv[1], ios::in);
    ofstream outputFile("Analisi", ios::app);
    if(!inputFile) {throw FileError();inputFile.close();exit(1);}
    inputFile>>problema.agenti;
    if(problema.agenti<=0){throw DataError();inputFile.close();exit(1);}
    inputFile>>problema.tipi;
    if(problema.tipi<=0){throw DataError();inputFile.close();exit(1);}
    inputFile>>problema.variabili;
    if(problema.variabili<=0){throw DataError();inputFile.close();exit(1);}
    inputFile>>problema.dominio;
    if(problema.dominio<=0){throw DataError();inputFile.close();exit(1);}
    IloNum densita;
    inputFile>>densita;
    if(densita<0 || densita>100) {throw DensityError();inputFile.close();exit(1);}
    IloInt vincoli =(IloInt)densita*problema.variabili*(problema.variabili-1)/200;
    problema.vincoli = vincoli;
    problema.Umin = IloNum(Random(-100.0, 0.0));
    problema.Umax = IloNum(Random(100, 1000));
    if(problema.Umax<=problema.Umin){
       throw UtilityError();
       inputFile.close();
       exit(1);}
    inputFile.close();
    IloInt vincolo = 1;
    if(argc < 3 ){cout<<"Per default il vincolo e' 1."<<endl;}
    else{
      char vin = *(argv[2]);
```

```
      if(IloInt(vin)< 48 || IloInt(vin)> 57){cout<<"Per default il vincolo e' 1."<<endl;}
      else{vincolo = IloInt(vin)-48;}}
    IloNum time, time_tot=0.0,time_medio;
    IloInt iter=0, optimal=0, out_time=0, infeasible=0, no_ris=0;
    for(int iteration=0; iteration<100; iteration++){
      IloInt righe = (IloInt)IloPower(problema.dominio,problema.variabili);
      IloInt colonne = problema.agenti*problema.tipi;
      IloNum ** Utilita;
      IloNum **  Probabilita;
      Utilita = new (IloNum *)[righe];
      Probabilita = new (IloNum *)[righe];
      for(IloInt i=0;i<righe;i++){
Utilita[i] = new IloNum[colonne];
Probabilita[i] = new IloNum[colonne];}
      for(IloInt i=0;i<righe;i++){for(IloInt j=0;j<colonne;j++){
Utilita[i][j]=0;
Probabilita[i][j]=0;}}
      CreaIstanzaProblema(env, problema, Utilita, Probabilita );
      IloNumVarArray var(env);
      IloRangeArray rng(env);
      IloObjective obj = IloMaximize(env);

      switch(vincolo){
      case 0:
AnalizzaVincoli(problema, Utilita, Probabilita);
iteration = 100;
break;
      case 1:
RandomExInterim(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      case 2:
RandomExPost(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      case 3:
RandomExInterimBNE(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      case 4:
RandomExPostDS(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      case 5:
DetExInterim(env, var, rng, obj, problema, Utilita, Probabilita);
break;
```

```
      case 6:
DetExPost(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      case 7:
DetExInterimBNE(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      case 8:
DetExPostDS(env, var, rng, obj, problema, Utilita, Probabilita);
break;
      default:
AnalizzaVincoli(problema, Utilita, Probabilita);
iteration = 100;
break;
      } // fine switch!!

      model.add(obj);
      model.add(rng);
      IloCplex cplex(model) ;
      IloTimer timer(env);

      cplex.exportModel("Model.lp");
      cplex.setParam(IloCplex:: RootAlg, IloCplex:: Primal);
      timer.start();
      if(!cplex.solve()){
time=timer.getTime();
timer.stop();
//...
no_ris++;
outputFile<<"Failed to optimize: ";
outputFile<<"Solution status = "<<cplex.getStatus()<<endl;
      }
      else{
time=timer.getTime();
timer.stop();
if(time>=0.36){
  outputFile<<"Fuori tempo."<<endl;
  out_time++;}
else{
  time_tot += time;
  iter++;

  outputFile<<"Solution status ="<<cplex.getStatus()<<endl;
```

```
  if(cplex.getStatus()==2){optimal++;}
  if(cplex.getStatus()==1){infeasible++;}
}
      }
      var.end();
      rng.end();
      obj.end();

    }// PARENTESI DI FOR
    time_medio=time_tot/iter;
    outputFile<<"RISULTATI: "<<endl<<endl;
    outputFile<<"Tempo medio impiegato: "<<time_medio<<endl;
    outputFile<<"Soluzioni ottimali: "<<optimal<<endl;
    outputFile<<"Senza risultato: "<<no_ris<<endl<<"fuori tempo: "<<out_time<<endl;
  } //parentesi di fine del try
  catch (IloException& ex){}
  catch (...){cerr<<"Unknown exception caught."<<endl;}
  //}// PARENTESI DI FOR
  env.end();
  return 0;
}
```

Esempio di modelli di programmazione lineare prodotti

In queste pagine sono raccolti i modelli di programmazione lineare risolti per ottenere i meccanismi descritti al capitolo 5. Riportiamo solo i modelli LP riguardanti i meccanismi deterministici, diversificandosi dai modelli riguardanti i meccanismi randomizzati solo per l'aggiunta della parte finale:

```
Generals
 id16  id17  id18  id19  id20 ...
```

che impone che le variabili specificate assumano solo valori interi.

L'istanza del problema

I parametri fissati sono:

- 2 agenti
- 3 tipi per ogni agente
- 3 variabili
- 2 valori nel dominio delle variabili
- una densità del problema del 100% (quindi valutazioni su 3 coppie)

La funzione *CreaIstanzaProblema* genera le matrici 5.14 e 5.15 contenenti le valutazioni dei due agenti.

	A_1, t_1	$A_1, t_2,$	A_1, t_3	A_2, t_1	$A_2, t_2,$	A_2, t_3
(0,0,0)	-9.267	206.369	319.886	48.326	163.856	357.971
(0,0,1)	55.74	211.727	282.082	80.575	178.412	342.258
(0,1,0)	22.346	172.503	346.827	24.645	144.664	297.575
(0,1,1)	46.326	195.195	316.198	26.818	156.072	336.524
(1,0,0)	22.239	194.773	276.324	0.317	204.57	318.366
(1,0,1)	61.311	194.123	294.06	35.387	186.468	317.35
(1,1,0)	0.508	152.92	322.572	6.922	177.408	280.027
(1,1,1)	-1.447	169.604	347.483	11.916	156.158	333.673

Tabella 5.14: Matrice rappresentante le valutazioni di 2 agenti per un sistema vincolato con 3 variabili.

	A_1, t_1	$A_1, t_2,$	A_1, t_3	A_2, t_1	$A_2, t_2,$	A_2, t_3
(0,0,0)	0.812	0.04	0.148	0.357	0.352	0.292
(0,0,1)	0.48	0.363	0.157	0.616	0.149	0.235
(0,1,0)	0.459	0.246	0.295	0.4	0.391	0.211
(0,1,1)	0.561	0.335	0.104	0.527	0.261	0.212
(1,0,0)	0.766	0.014	0.219	0.396	0.416	0.187
(1,0,1)	0.594	0.251	0.155	0.815	0.149	0.035
(1,1,0)	0.417	0.159	0.424	0.228	0.454	0.318
(1,1,1)	0.679	0.162	0.16	0.515	0.26	0.224

Tabella 5.15: Matrice delle probabilità con cui ogni agente riporta ciascun tipo, in funzione alla tupla cui si riferisce.

Meccanismo Deterministico, Ex interim IR

```
\Problem name: DetExInterim.lp

Maximize
 obj: 11.323 id16 + 40.306 id17 + 8.627 id18 + 21.625 id19 + 6.842 id20
 + 46.812 id21 + 0.706 id22 + 3.661 id23 + 44.185 id24 + 16.747 id25
 + 29.973 id26 + 29.635 id27 + 72.274 id28 + 21.93 id29 + 33.683 id30
 + 27.313 id31 + 82.679 id32 + 44.894 id33 + 30.984 id34 + 45.533 id35
 + 48.789 id36 + 7.872 id37 + 37.201 id38 + 50.53 id39 + 3.637 id40
 + 65.361 id41 + 19.399 id42 + 39.195 id43 + 1.082 id44 + 46.95 id45
 + 5.795 id46 + 15.144 id47 + 5.213 id48 + 21.101 id49 + 30.507 id50
 + 30.713 id51 + 2.326 id52 + 14.234 id53 + 23.845 id54 + 13.721 id55
```

```
 + 6.591 id56 + 47.258 id57 + 24.4 id58 + 37.763 id59 + 1.343 id60
 + 4.493 id61 + 21.891 id62 + 18.263 id63 + 19.455 id64 + 35.073 id65
 + 43.834 id66 + 18.8 id67 + 23.991 id68 + 41.617 id69 + 31.853 id70
 + 29.614 id71 + 25.201 id72 + 10.772 id73 + 56.691 id74 + 12.819 id75
 + 43.811 id76 + 11.098 id77 + 96.244 id78 + 20.951 id79 + 29.294 id80
 + 23.035 id81 + 40.111 id82 + 14.391 id83 + 24.354 id84 + 3.317 id85
 + 81.25 id86 + 24.413 id87
Subject To
 id1:  - 3.308 id16 + 34.336 id17 + 8.938 id18 + 24.414 id19 + 8.807 id20
       + 49.968 id21 + 0.116 id22 - 0.745 id23 - 3.261 id24 + 8.305 id25
       + 8.737 id26 + 12.091 id27 + 9.251 id28 + 9.135 id29 + 0.231 id30
       - 0.376 id31 - 2.705 id32 + 13.099 id33 + 4.715 id34 + 9.821 id35
       + 4.159 id36 + 2.146 id37 + 0.161 id38 - 0.324 id39 >= 0
 id2:  73.674 id40 + 130.424 id41 + 69.001 id42 + 102.868 id43 + 77.13 id44
       + 158.21 id45 + 34.866 id46 + 87.346 id47 + 72.642 id48 + 31.547 id49
       + 67.449 id50 + 50.946 id51 + 81.026 id52 + 28.924 id53 + 69.426 id54
       + 44.097 id55 + 60.26 id56 + 49.756 id57 + 36.398 id58 + 41.381 id59
       + 36.423 id60 + 6.794 id61 + 48.629 id62 + 37.991 id63 >= 0
 id3:  114.199 id64 + 173.762 id65 + 138.731 id66 + 166.636 id67 + 109.424 id68
       + 239.659 id69 + 73.546 id70 + 178.954 id71 + 112.6 id72 + 42.03 id73
       + 135.609 id74 + 82.528 id75 + 114.951 id76 + 43.815 id77
       + 146.448 id78 + 90.346 id79 + 93.407 id80 + 66.289 id81 + 73.18 id82
       + 67.034 id83 + 51.673 id84 + 10.292 id85 + 102.578 id86 + 77.836 id87
       >= 0
 id4:  39.241 id16 + 38.676 id17 + 11.312 id18 + 15.045 id19 + 0.243 id20
       + 21.02 id21 + 2.886 id22 + 8.091 id23 + 1.933 id40 + 29.249 id41
       + 6.063 id42 + 8.984 id43 + 0.004 id44 + 8.882 id45 + 1.101 id46
       + 1.93 id47 + 7.152 id64 + 12.65 id65 + 7.27 id66 + 2.789 id67
       + 0.069 id68 + 5.485 id69 + 2.935 id70 + 1.907 id71 >= 0
 id5:  133.051 id24 + 85.638 id25 + 66.401 id26 + 87.556 id27 + 156.701 id28
       + 110.762 id29 + 73.979 id30 + 106.031 id31 + 6.554 id48 + 64.764 id49
       + 35.587 id50 + 52.284 id51 + 2.864 id52 + 46.803 id53 + 28.208 id54
       + 25.298 id55 + 24.251 id72 + 28.011 id73 + 42.676 id74 + 16.231 id75
       + 44.801 id76 + 28.902 id77 + 75.221 id78 + 24.985 id79 >= 0
 id6:  290.672 id32 + 164.284 id33 + 136.587 id34 + 188.79 id35 + 243.868 id36
       + 188.506 id37 + 116.771 id38 + 226.564 id39 + 14.319 id56
       + 124.24 id57 + 73.203 id58 + 112.735 id59 + 4.457 id60 + 79.655 id61
       + 44.524 id62 + 54.055 id63 + 52.98 id80 + 53.734 id81 + 87.785 id82
       + 34.998 id83 + 69.722 id84 + 49.189 id85 + 118.731 id86 + 53.388 id87
       >= 0
 id7:  id16 + id17 + id18 + id19 + id20 + id21 + id22 + id23  = 1
 id8:  id24 + id25 + id26 + id27 + id28 + id29 + id30 + id31  = 1
```

```
 id9:  id32 + id33 + id34 + id35 + id36 + id37 + id38 + id39  = 1
 id10: id40 + id41 + id42 + id43 + id44 + id45 + id46 + id47  = 1
 id11: id48 + id49 + id50 + id51 + id52 + id53 + id54 + id55  = 1
 id12: id56 + id57 + id58 + id59 + id60 + id61 + id62 + id63  = 1
 id13: id64 + id65 + id66 + id67 + id68 + id69 + id70 + id71  = 1
 id14: id72 + id73 + id74 + id75 + id76 + id77 + id78 + id79  = 1
 id15: id80 + id81 + id82 + id83 + id84 + id85 + id86 + id87  = 1
Bounds
 0 <= id16 <= 1   ...    0 <= id87 <= 1
Generals
 id16  id17  id18  id19  id20  id21  id22  id23  id24  id25  id26  id27  id28
 id29  id30  id31  id32  id33  id34  id35  id36  id37  id38  id39  id40  id41
 id42  id43  id44  id45  id46  id47  id48  id49  id50  id51  id52  id53  id54
 id55  id56  id57  id58  id59  id60  id61  id62  id63  id64  id65  id66  id67
 id68  id69  id70  id71  id72  id73  id74  id75  id76  id77  id78  id79  id80
 id81  id82  id83  id84  id85  id86  id87
End
```

Meccanismo Deterministico, Ex post IR

```
\Problem name: DetExPost.lp

Maximize
 obj: 11.323 id28 + 40.306 id29 + 8.627 id30 + 21.625 id31 + 6.842 id32
 + 46.812 id33 + 0.706 id34 + 3.661 id35 + 44.185 id36 + 16.747 id37
 + 29.973 id38 + 29.635 id39 + 72.274 id40 + 21.93 id41 + 33.683 id42
 + 27.313 id43 + 82.679 id44 + 44.894 id45 + 30.984 id46 + 45.533 id47
 + 48.789 id48 + 7.872 id49 + 37.201 id50 + 50.53 id51 + 3.637 id52
 + 65.361 id53 + 19.399 id54 + 39.195 id55 + 1.082 id56 + 46.95 id57
 + 5.795 id58 + 15.144 id59 + 5.213 id60 + 21.101 id61 + 30.507 id62
 + 30.713 id63 + 2.326 id64 + 14.234 id65 + 23.845 id66 + 13.721 id67
 + 6.591 id68 + 47.258 id69 + 24.4 id70 + 37.763 id71 + 1.343 id72
 + 4.493 id73 + 21.891 id74 + 18.263 id75 + 19.455 id76 + 35.073 id77
 + 43.834 id78 + 18.8 id79 + 23.991 id80 + 41.617 id81 + 31.853 id82
 + 29.614 id83 + 25.201 id84 + 10.772 id85 + 56.691 id86 + 12.819 id87
 + 43.811 id88 + 11.098 id89 + 96.244 id90 + 20.951 id91 + 29.294 id92
 + 23.035 id93 + 40.111 id94 + 14.391 id95 + 24.354 id96 + 3.317 id97
 + 81.25 id98 + 24.413 id99
Subject To
 id1:  - 9.267 id28 + 55.74 id29 + 22.346 id30 + 46.326 id31 + 22.239 id32
       + 61.311 id33 + 0.508 id34 - 1.447 id35 >= 0
```

```
id2:  48.326 id28 + 80.575 id29 + 24.645 id30 + 26.818 id31 + 0.317 id32
      + 35.387 id33 + 6.922 id34 + 11.916 id35 >= 0
id3:  - 9.267 id36 + 55.74 id37 + 22.346 id38 + 46.326 id39 + 22.239 id40
      + 61.311 id41 + 0.508 id42 - 1.447 id43 >= 0
id4:  163.856 id36 + 178.412 id37 + 144.664 id38 + 156.072 id39 + 204.57 id40
      + 186.468 id41 + 177.408 id42 + 156.158 id43 >= 0
id5:  - 9.267 id44 + 55.74 id45 + 22.346 id46 + 46.326 id47 + 22.239 id48
      + 61.311 id49 + 0.508 id50 - 1.447 id51 >= 0
id6:  357.971 id44 + 342.258 id45 + 297.575 id46 + 336.524 id47 + 318.366 id48
      + 317.35 id49 + 280.027 id50 + 333.673 id51 >= 0
id7:  206.369 id52 + 211.727 id53 + 172.503 id54 + 195.195 id55 + 194.773 id56
      + 194.123 id57 + 152.92 id58 + 169.604 id59 >= 0
id8:  48.326 id52 + 80.575 id53 + 24.645 id54 + 26.818 id55 + 0.317 id56
      + 35.387 id57 + 6.922 id58 + 11.916 id59 >= 0
id9:  206.369 id60 + 211.727 id61 + 172.503 id62 + 195.195 id63 + 194.773 id64
      + 194.123 id65 + 152.92 id66 + 169.604 id67 >= 0
id10: 163.856 id60 + 178.412 id61 + 144.664 id62 + 156.072 id63 + 204.57 id64
      + 186.468 id65 + 177.408 id66 + 156.158 id67 >= 0
id11: 206.369 id68 + 211.727 id69 + 172.503 id70 + 195.195 id71 + 194.773 id72
      + 194.123 id73 + 152.92 id74 + 169.604 id75 >= 0
id12: 357.971 id68 + 342.258 id69 + 297.575 id70 + 336.524 id71 + 318.366 id72
      + 317.35 id73 + 280.027 id74 + 333.673 id75 >= 0
id13: 319.886 id76 + 282.082 id77 + 346.827 id78 + 316.198 id79 + 276.324 id80
      + 294.06 id81 + 322.572 id82 + 347.483 id83 >= 0
id14: 48.326 id76 + 80.575 id77 + 24.645 id78 + 26.818 id79 + 0.317 id80
      + 35.387 id81 + 6.922 id82 + 11.916 id83 >= 0
id15: 319.886 id84 + 282.082 id85 + 346.827 id86 + 316.198 id87 + 276.324 id88
      + 294.06 id89 + 322.572 id90 + 347.483 id91 >= 0
id16: 163.856 id84 + 178.412 id85 + 144.664 id86 + 156.072 id87 + 204.57 id88
      + 186.468 id89 + 177.408 id90 + 156.158 id91 >= 0
id17: 319.886 id92 + 282.082 id93 + 346.827 id94 + 316.198 id95 + 276.324 id96
      + 294.06 id97 + 322.572 id98 + 347.483 id99 >= 0
id18: 357.971 id92 + 342.258 id93 + 297.575 id94 + 336.524 id95 + 318.366 id96
      + 317.35 id97 + 280.027 id98 + 333.673 id99 >= 0
id19: id28 + id29 + id30 + id31 + id32 + id33 + id34 + id35  = 1
id20: id36 + id37 + id38 + id39 + id40 + id41 + id42 + id43  = 1
id21: id44 + id45 + id46 + id47 + id48 + id49 + id50 + id51  = 1
id22: id52 + id53 + id54 + id55 + id56 + id57 + id58 + id59  = 1
id23: id60 + id61 + id62 + id63 + id64 + id65 + id66 + id67  = 1
id24: id68 + id69 + id70 + id71 + id72 + id73 + id74 + id75  = 1
id25: id76 + id77 + id78 + id79 + id80 + id81 + id82 + id83  = 1
id26: id84 + id85 + id86 + id87 + id88 + id89 + id90 + id91  = 1
```

```
 id27: id92 + id93 + id94 + id95 + id96 + id97 + id98 + id99  = 1
Bounds
 0 <= id28 <= 1   ...   0 <= id99 <= 1
Generals
 id28  id29  id30  id31  id32  id33  id34  id35  id36  id37  id38  id39  id40
 id41  id42  id43  id44  id45  id46  id47  id48  id49  id50  id51  id52  id53
 id54  id55  id56  id57  id58  id59  id60  id61  id62  id63  id64  id65  id66
 id67  id68  id69  id70  id71  id72  id73  id74  id75  id76  id77  id78  id79
 id80  id81  id82  id83  id84  id85  id86  id87  id88  id89  id90  id91  id92
 id93  id94  id95  id96  id97  id98  id99
End
```

Meccanismo Deterministico, Ex interim IR, implementato secondo gli equilibri di Bayes-Nash

```
\Problem name: DetExInterimBNE.lp

Maximize
 obj: 11.323 id16 + 40.306 id17 + 8.627 id18 + 21.625 id19 + 6.842 id20
 + 46.812 id21 + 0.706 id22 + 3.661 id23 + 44.185 id24 + 16.747 id25
 + 29.973 id26 + 29.635 id27 + 72.274 id28 + 21.93 id29 + 33.683 id30
 + 27.313 id31 + 82.679 id32 + 44.894 id33 + 30.984 id34 + 45.533 id35
 + 48.789 id36 + 7.872 id37 + 37.201 id38 + 50.53 id39 + 3.637 id40
 + 65.361 id41 + 19.399 id42 + 39.195 id43 + 1.082 id44 + 46.95 id45
 + 5.795 id46 + 15.144 id47 + 5.213 id48 + 21.101 id49 + 30.507 id50
 + 30.713 id51 + 2.326 id52 + 14.234 id53 + 23.845 id54 + 13.721 id55
 + 6.591 id56 + 47.258 id57 + 24.4 id58 + 37.763 id59 + 1.343 id60
 + 4.493 id61 + 21.891 id62 + 18.263 id63 + 19.455 id64 + 35.073 id65
 + 43.834 id66 + 18.8 id67 + 23.991 id68 + 41.617 id69 + 31.853 id70
 + 29.614 id71 + 25.201 id72 + 10.772 id73 + 56.691 id74 + 12.819 id75
 + 43.811 id76 + 11.098 id77 + 96.244 id78 + 20.951 id79 + 29.294 id80
 + 23.035 id81 + 40.111 id82 + 14.391 id83 + 24.354 id84 + 3.317 id85
 + 81.25 id86 + 24.413 id87
Subject To
 id1:   - 3.308 id16 + 34.336 id17 + 8.938 id18 + 24.414 id19 + 8.807 id20
        + 49.968 id21 + 0.116 id22 - 0.745 id23 - 3.261 id24 + 8.305 id25
        + 8.737 id26 + 12.091 id27 + 9.251 id28 + 9.135 id29 + 0.231 id30
        - 0.376 id31 - 2.705 id32 + 13.099 id33 + 4.715 id34 + 9.821 id35
        + 4.159 id36 + 2.146 id37 + 0.161 id38 - 0.324 id39 >= 0
 id2:   73.674 id40 + 130.424 id41 + 69.001 id42 + 102.868 id43 + 77.13 id44
```

```
       + 158.21 id45 + 34.866 id46 + 87.346 id47 + 72.642 id48 + 31.547 id49
       + 67.449 id50 + 50.946 id51 + 81.026 id52 + 28.924 id53 + 69.426 id54
       + 44.097 id55 + 60.26 id56 + 49.756 id57 + 36.398 id58 + 41.381 id59
       + 36.423 id60 + 6.794 id61 + 48.629 id62 + 37.991 id63 >= 0
id3:   114.199 id64 + 173.762 id65 + 138.731 id66 + 166.636 id67
       + 109.424 id68 + 239.659 id69 + 73.546 id70 + 178.954 id71
       + 112.6 id72 + 42.03 id73 + 135.609 id74 + 82.528 id75 + 114.951 id76
       + 43.815 id77 + 146.448 id78 + 90.346 id79 + 93.407 id80 + 66.289 id81
       + 73.18 id82 + 67.034 id83 + 51.673 id84 + 10.292 id85 + 102.578 id86
       + 77.836 id87 >= 0
id4:   39.241 id16 + 38.676 id17 + 11.312 id18 + 15.045 id19 + 0.243 id20
       + 21.02 id21 + 2.886 id22 + 8.091 id23 + 1.933 id40 + 29.249 id41
       + 6.063 id42 + 8.984 id43 + 0.004 id44 + 8.882 id45 + 1.101 id46
       + 1.93 id47 + 7.152 id64 + 12.65 id65 + 7.27 id66 + 2.789 id67
       + 0.069 id68 + 5.485 id69 + 2.935 id70 + 1.907 id71 >= 0
id5:   133.051 id24 + 85.638 id25 + 66.401 id26 + 87.556 id27 + 156.701 id28
       + 110.762 id29 + 73.979 id30 + 106.031 id31 + 6.554 id48 + 64.764 id49
       + 35.587 id50 + 52.284 id51 + 2.864 id52 + 46.803 id53 + 28.208 id54
       + 25.298 id55 + 24.251 id72 + 28.011 id73 + 42.676 id74 + 16.231 id75
       + 44.801 id76 + 28.902 id77 + 75.221 id78 + 24.985 id79 >= 0
id6:   290.672 id32 + 164.284 id33 + 136.587 id34 + 188.79 id35 + 243.868 id36
       + 188.506 id37 + 116.771 id38 + 226.564 id39 + 14.319 id56
       + 124.24 id57 + 73.203 id58 + 112.735 id59 + 4.457 id60 + 79.655 id61
       + 44.524 id62 + 54.055 id63 + 52.98 id80 + 53.734 id81 + 87.785 id82
       + 34.998 id83 + 69.722 id84 + 49.189 id85 + 118.731 id86 + 53.388 id87
       >= 0
id7:   id16 + id17 + id18 + id19 + id20 + id21 + id22 + id23  = 1
id8:   id24 + id25 + id26 + id27 + id28 + id29 + id30 + id31  = 1
id9:   id32 + id33 + id34 + id35 + id36 + id37 + id38 + id39  = 1
id10:  id40 + id41 + id42 + id43 + id44 + id45 + id46 + id47  = 1
id11:  id48 + id49 + id50 + id51 + id52 + id53 + id54 + id55  = 1
id12:  id56 + id57 + id58 + id59 + id60 + id61 + id62 + id63  = 1
id13:  id64 + id65 + id66 + id67 + id68 + id69 + id70 + id71  = 1
id14:  id72 + id73 + id74 + id75 + id76 + id77 + id78 + id79  = 1
id15:  id80 + id81 + id82 + id83 + id84 + id85 + id86 + id87  = 1
id88:  - 3.308319 id16 + 34.33584 id17 + 8.9384 id18 + 24.413802 id19
       + 8.806644 id20 + 49.968465 id21 + 0.115824 id22 - 0.745205 id23
       + 3.308319 id40 - 34.33584 id41 - 8.9384 id42 - 24.413802 id43
       - 8.806644 id44 - 49.968465 id45 - 0.115824 id46 + 0.745205 id47 >= 0
id89:  - 3.308319 id16 + 34.33584 id17 + 8.9384 id18 + 24.413802 id19
       + 8.806644 id20 + 49.968465 id21 + 0.115824 id22 - 0.745205 id23
       + 3.308319 id64 - 34.33584 id65 - 8.9384 id66 - 24.413802 id67
```

```
       - 8.806644 id68 - 49.968465 id69 - 0.115824 id70 + 0.745205 id71 >= 0
id90:  - 3.261984 id24 + 8.30526 id25 + 8.737286 id26 + 12.091086 id27
       + 9.251424 id28 + 9.135339 id29 + 0.230632 id30 - 0.37622 id31
       + 3.261984 id48 - 8.30526 id49 - 8.737286 id50 - 12.091086 id51
       - 9.251424 id52 - 9.135339 id53 - 0.230632 id54 + 0.37622 id55 >= 0
id91:  - 3.261984 id24 + 8.30526 id25 + 8.737286 id26 + 12.091086 id27
       + 9.251424 id28 + 9.135339 id29 + 0.230632 id30 - 0.37622 id31
       + 3.261984 id72 - 8.30526 id73 - 8.737286 id74 - 12.091086 id75
       - 9.251424 id76 - 9.135339 id77 - 0.230632 id78 + 0.37622 id79 >= 0
id92:  - 2.705964 id32 + 13.0989 id33 + 4.715006 id34 + 9.821112 id35
       + 4.158693 id36 + 2.145885 id37 + 0.161544 id38 - 0.324128 id39
       + 2.705964 id56 - 13.0989 id57 - 4.715006 id58 - 9.821112 id59
       - 4.158693 id60 - 2.145885 id61 - 0.161544 id62 + 0.324128 id63 >= 0
id93:  - 2.705964 id32 + 13.0989 id33 + 4.715006 id34 + 9.821112 id35
       + 4.158693 id36 + 2.145885 id37 + 0.161544 id38 - 0.324128 id39
       + 2.705964 id80 - 13.0989 id81 - 4.715006 id82 - 9.821112 id83
       - 4.158693 id84 - 2.145885 id85 - 0.161544 id86 + 0.324128 id87 >= 0
id94:  - 73.673733 id16 - 130.423832 id17 - 69.0012 id18 - 102.867765 id19
       - 77.130108 id20 - 158.210245 id21 - 34.86576 id22 - 87.34606 id23
       + 73.673733 id40 + 130.423832 id41 + 69.0012 id42 + 102.867765 id43
       + 77.130108 id44 + 158.210245 id45 + 34.86576 id46 + 87.34606 id47
       >= 0
id95:  73.673733 id40 + 130.423832 id41 + 69.0012 id42 + 102.867765 id43
       + 77.130108 id44 + 158.210245 id45 + 34.86576 id46 + 87.34606 id47
       - 73.673733 id64 - 130.423832 id65 - 69.0012 id66 - 102.867765 id67
       - 77.130108 id68 - 158.210245 id69 - 34.86576 id70 - 87.34606 id71
       >= 0
id96:  - 72.641888 id24 - 31.547323 id25 - 67.448673 id26 - 50.945895 id27
       - 81.025568 id28 - 28.924327 id29 - 69.42568 id30 - 44.09704 id31
       + 72.641888 id48 + 31.547323 id49 + 67.448673 id50 + 50.945895 id51
       + 81.025568 id52 + 28.924327 id53 + 69.42568 id54 + 44.09704 id55 >= 0
id97:  72.641888 id48 + 31.547323 id49 + 67.448673 id50 + 50.945895 id51
       + 81.025568 id52 + 28.924327 id53 + 69.42568 id54 + 44.09704 id55
       - 72.641888 id72 - 31.547323 id73 - 67.448673 id74 - 50.945895 id75
       - 81.025568 id76 - 28.924327 id77 - 69.42568 id78 - 44.09704 id79 >= 0
id98:  - 60.259748 id32 - 49.755845 id33 - 36.398133 id34 - 41.38134 id35
       - 36.422551 id36 - 6.794305 id37 - 48.62856 id38 - 37.991296 id39
       + 60.259748 id56 + 49.755845 id57 + 36.398133 id58 + 41.38134 id59
       + 36.422551 id60 + 6.794305 id61 + 48.62856 id62 + 37.991296 id63 >= 0
id99:  60.259748 id56 + 49.755845 id57 + 36.398133 id58 + 41.38134 id59
       + 36.422551 id60 + 6.794305 id61 + 48.62856 id62 + 37.991296 id63
       - 60.259748 id80 - 49.755845 id81 - 36.398133 id82 - 41.38134 id83
```

```
       - 36.422551 id84 - 6.794305 id85 - 48.62856 id86 - 37.991296 id87 >= 0
id100: - 114.199302 id16 - 173.762512 id17 - 138.7308 id18 - 166.636346 id19
       - 109.424304 id20 - 239.6589 id21 - 73.546416 id22 - 178.953745 id23
       + 114.199302 id64 + 173.762512 id65 + 138.7308 id66 + 166.636346 id67
       + 109.424304 id68 + 239.6589 id69 + 73.546416 id70 + 178.953745 id71
       >= 0
id101: - 114.199302 id40 - 173.762512 id41 - 138.7308 id42 - 166.636346 id43
       - 109.424304 id44 - 239.6589 id45 - 73.546416 id46 - 178.953745 id47
       + 114.199302 id64 + 173.762512 id65 + 138.7308 id66 + 166.636346 id67
       + 109.424304 id68 + 239.6589 id69 + 73.546416 id70 + 178.953745 id71
       >= 0
id102: - 112.599872 id24 - 42.030218 id25 - 135.609357 id26 - 82.527678 id27
       - 114.950784 id28 - 43.81494 id29 - 146.447688 id30 - 90.34558 id31
       + 112.599872 id72 + 42.030218 id73 + 135.609357 id74 + 82.527678 id75
       + 114.950784 id76 + 43.81494 id77 + 146.447688 id78 + 90.34558 id79
       >= 0
id103: - 112.599872 id48 - 42.030218 id49 - 135.609357 id50 - 82.527678 id51
       - 114.950784 id52 - 43.81494 id53 - 146.447688 id54 - 90.34558 id55
       + 112.599872 id72 + 42.030218 id73 + 135.609357 id74 + 82.527678 id75
       + 114.950784 id76 + 43.81494 id77 + 146.447688 id78 + 90.34558 id79
       >= 0
id104: - 93.406712 id32 - 66.28927 id33 - 73.180497 id34 - 67.033976 id35
       - 51.672588 id36 - 10.2921 id37 - 102.577896 id38 - 77.836192 id39
       + 93.406712 id80 + 66.28927 id81 + 73.180497 id82 + 67.033976 id83
       + 51.672588 id84 + 10.2921 id85 + 102.577896 id86 + 77.836192 id87
       >= 0
id105: - 93.406712 id56 - 66.28927 id57 - 73.180497 id58 - 67.033976 id59
       - 51.672588 id60 - 10.2921 id61 - 102.577896 id62 - 77.836192 id63
       + 93.406712 id80 + 66.28927 id81 + 73.180497 id82 + 67.033976 id83
       + 51.672588 id84 + 10.2921 id85 + 102.577896 id86 + 77.836192 id87
       >= 0
id106: 39.240712 id16 + 38.676 id17 + 11.312055 id18 + 15.044898 id19
       + 0.242822 id20 + 21.019878 id21 + 2.886474 id22 + 8.090964 id23
       - 39.240712 id24 - 38.676 id25 - 11.312055 id26 - 15.044898 id27
       - 0.242822 id28 - 21.019878 id29 - 2.886474 id30 - 8.090964 id31 >= 0
id107: 39.240712 id16 + 38.676 id17 + 11.312055 id18 + 15.044898 id19
       + 0.242822 id20 + 21.019878 id21 + 2.886474 id22 + 8.090964 id23
       - 39.240712 id32 - 38.676 id33 - 11.312055 id34 - 15.044898 id35
       - 0.242822 id36 - 21.019878 id37 - 2.886474 id38 - 8.090964 id39 >= 0
id108: 1.93304 id40 + 29.248725 id41 + 6.06267 id42 + 8.98403 id43
       + 0.004438 id44 + 8.882137 id45 + 1.100598 id46 + 1.930392 id47
       - 1.93304 id48 - 29.248725 id49 - 6.06267 id50 - 8.98403 id51
```

```
       - 0.004438 id52 - 8.882137 id53 - 1.100598 id54 - 1.930392 id55 >= 0
id109: 1.93304 id40 + 29.248725 id41 + 6.06267 id42 + 8.98403 id43
       + 0.004438 id44 + 8.882137 id45 + 1.100598 id46 + 1.930392 id47
       - 1.93304 id56 - 29.248725 id57 - 6.06267 id58 - 8.98403 id59
       - 0.004438 id60 - 8.882137 id61 - 1.100598 id62 - 1.930392 id63 >= 0
id110: 7.152248 id64 + 12.650275 id65 + 7.270275 id66 + 2.789072 id67
       + 0.069423 id68 + 5.484985 id69 + 2.934928 id70 + 1.90656 id71
       - 7.152248 id72 - 12.650275 id73 - 7.270275 id74 - 2.789072 id75
       - 0.069423 id76 - 5.484985 id77 - 2.934928 id78 - 1.90656 id79 >= 0
id111: 7.152248 id64 + 12.650275 id65 + 7.270275 id66 + 2.789072 id67
       + 0.069423 id68 + 5.484985 id69 + 2.934928 id70 + 1.90656 id71
       - 7.152248 id80 - 12.650275 id81 - 7.270275 id82 - 2.789072 id83
       - 0.069423 id84 - 5.484985 id85 - 2.934928 id86 - 1.90656 id87 >= 0
id112: - 133.051072 id16 - 85.63776 id17 - 66.400776 id18 - 87.556392 id19
       - 156.70062 id20 - 110.761992 id21 - 73.979136 id22 - 106.031282 id23
       + 133.051072 id24 + 85.63776 id25 + 66.400776 id26 + 87.556392 id27
       + 156.70062 id28 + 110.761992 id29 + 73.979136 id30 + 106.031282 id31
       >= 0
id113: 133.051072 id24 + 85.63776 id25 + 66.400776 id26 + 87.556392 id27
       + 156.70062 id28 + 110.761992 id29 + 73.979136 id30 + 106.031282 id31
       - 133.051072 id32 - 85.63776 id33 - 66.400776 id34 - 87.556392 id35
       - 156.70062 id36 - 110.761992 id37 - 73.979136 id38 - 106.031282 id39
       >= 0
id114: - 6.55424 id40 - 64.763556 id41 - 35.587344 id42 - 52.28412 id43
       - 2.86398 id44 - 46.803468 id45 - 28.207872 id46 - 25.297596 id47
       + 6.55424 id48 + 64.763556 id49 + 35.587344 id50 + 52.28412 id51
       + 2.86398 id52 + 46.803468 id53 + 28.207872 id54 + 25.297596 id55 >= 0
id115: 6.55424 id48 + 64.763556 id49 + 35.587344 id50 + 52.28412 id51
       + 2.86398 id52 + 46.803468 id53 + 28.207872 id54 + 25.297596 id55
       - 6.55424 id56 - 64.763556 id57 - 35.587344 id58 - 52.28412 id59
       - 2.86398 id60 - 46.803468 id61 - 28.207872 id62 - 25.297596 id63 >= 0
id116: - 24.250688 id64 - 28.010684 id65 - 42.67588 id66 - 16.231488 id67
       - 44.80083 id68 - 28.90254 id69 - 75.220992 id70 - 24.98528 id71
       + 24.250688 id72 + 28.010684 id73 + 42.67588 id74 + 16.231488 id75
       + 44.80083 id76 + 28.90254 id77 + 75.220992 id78 + 24.98528 id79 >= 0
id117: 24.250688 id72 + 28.010684 id73 + 42.67588 id74 + 16.231488 id75
       + 44.80083 id76 + 28.90254 id77 + 75.220992 id78 + 24.98528 id79
       - 24.250688 id80 - 28.010684 id81 - 42.67588 id82 - 16.231488 id83
       - 44.80083 id84 - 28.90254 id85 - 75.220992 id86 - 24.98528 id87 >= 0
id118: - 290.672452 id16 - 164.28384 id17 - 136.586925 id18 - 188.789964 id19
       - 243.868356 id20 - 188.5059 id21 - 116.771259 id22 - 226.563967 id23
       + 290.672452 id32 + 164.28384 id33 + 136.586925 id34 + 188.789964 id35
```

```
        + 243.868356 id36 + 188.5059 id37 + 116.771259 id38 + 226.563967 id39
        >= 0
 id119: - 290.672452 id24 - 164.28384 id25 - 136.586925 id26 - 188.789964 id27
        - 243.868356 id28 - 188.5059 id29 - 116.771259 id30 - 226.563967 id31
        + 290.672452 id32 + 164.28384 id33 + 136.586925 id34 + 188.789964 id35
        + 243.868356 id36 + 188.5059 id37 + 116.771259 id38 + 226.563967 id39
        >= 0
 id120: - 14.31884 id40 - 124.239654 id41 - 73.20345 id42 - 112.73554 id43
        - 4.457124 id44 - 79.65485 id45 - 44.524293 id46 - 54.055026 id47
        + 14.31884 id56 + 124.239654 id57 + 73.20345 id58 + 112.73554 id59
        + 4.457124 id60 + 79.65485 id61 + 44.524293 id62 + 54.055026 id63 >= 0
 id121: - 14.31884 id48 - 124.239654 id49 - 73.20345 id50 - 112.73554 id51
        - 4.457124 id52 - 79.65485 id53 - 44.524293 id54 - 54.055026 id55
        + 14.31884 id56 + 124.239654 id57 + 73.20345 id58 + 112.73554 id59
        + 4.457124 id60 + 79.65485 id61 + 44.524293 id62 + 54.055026 id63 >= 0
 id122: - 52.979708 id64 - 53.734506 id65 - 87.784625 id66 - 34.998496 id67
        - 69.722154 id68 - 49.18925 id69 - 118.731448 id70 - 53.38768 id71
        + 52.979708 id80 + 53.734506 id81 + 87.784625 id82 + 34.998496 id83
        + 69.722154 id84 + 49.18925 id85 + 118.731448 id86 + 53.38768 id87
        >= 0
 id123: - 52.979708 id72 - 53.734506 id73 - 87.784625 id74 - 34.998496 id75
        - 69.722154 id76 - 49.18925 id77 - 118.731448 id78 - 53.38768 id79
        + 52.979708 id80 + 53.734506 id81 + 87.784625 id82 + 34.998496 id83
        + 69.722154 id84 + 49.18925 id85 + 118.731448 id86 + 53.38768 id87
        >= 0
Bounds
 0 <= id16 <= 1   ...   0 <= id87 <= 1
Generals
 id16  id17  id18  id19  id20  id21  id22  id23  id24  id25  id26  id27  id28
 id29  id30  id31  id32  id33  id34  id35  id36  id37  id38  id39  id40  id41
 id42  id43  id44  id45  id46  id47  id48  id49  id50  id51  id52  id53  id54
 id55  id56  id57  id58  id59  id60  id61  id62  id63  id64  id65  id66  id67
 id68  id69  id70  id71  id72  id73  id74  id75  id76  id77  id78  id79  id80
 id81  id82  id83  id84  id85  id86  id87
End
```

Meccanismo Deterministico, Ex post IR, implementato secondo le strategie dominanti

```
 \Problem name: DetExPostDS.lp
```

```
Maximize
 obj: 11.323 id64 + 40.306 id65 + 8.627 id66 + 21.625 id67 + 6.842 id68
 + 46.812 id69 + 0.706 id70 + 3.661 id71 + 44.185 id72 + 16.747 id73
 + 29.973 id74 + 29.635 id75 + 72.274 id76 + 21.93 id77 + 33.683 id78
 + 27.313 id79 + 82.679 id80 + 44.894 id81 + 30.984 id82 + 45.533 id83
 + 48.789 id84 + 7.872 id85 + 37.201 id86 + 50.53 id87 + 3.637 id88
 + 65.361 id89 + 19.399 id90 + 39.195 id91 + 1.082 id92 + 46.95 id93
 + 5.795 id94 + 15.144 id95 + 5.213 id96 + 21.101 id97 + 30.507 id98
 + 30.713 id99 + 2.326 id100 + 14.234 id101 + 23.845 id102 + 13.721 id103
 + 6.591 id104 + 47.258 id105 + 24.4 id106 + 37.763 id107 + 1.343 id108
 + 4.493 id109 + 21.891 id110 + 18.263 id111 + 19.455 id112 + 35.073 id113
 + 43.834 id114 + 18.8 id115 + 23.991 id116 + 41.617 id117 + 31.853 id118
 + 29.614 id119 + 25.201 id120 + 10.772 id121 + 56.691 id122 + 12.819 id123
 + 43.811 id124 + 11.098 id125 + 96.244 id126 + 20.951 id127 + 29.294 id128
 + 23.035 id129 + 40.111 id130 + 14.391 id131 + 24.354 id132 + 3.317 id133
 + 81.25 id134 + 24.413 id135
Subject To
 id1:  - 9.267 id64 + 55.74 id65 + 22.346 id66 + 46.326 id67 + 22.239 id68
       + 61.311 id69 + 0.508 id70 - 1.447 id71 >= 0
 id2:  48.326 id64 + 80.575 id65 + 24.645 id66 + 26.818 id67 + 0.317 id68
       + 35.387 id69 + 6.922 id70 + 11.916 id71 >= 0
 id3:  - 9.267 id72 + 55.74 id73 + 22.346 id74 + 46.326 id75 + 22.239 id76
       + 61.311 id77 + 0.508 id78 - 1.447 id79 >= 0
 id4:  163.856 id72 + 178.412 id73 + 144.664 id74 + 156.072 id75 + 204.57 id76
       + 186.468 id77 + 177.408 id78 + 156.158 id79 >= 0
 id5:  - 9.267 id80 + 55.74 id81 + 22.346 id82 + 46.326 id83 + 22.239 id84
       + 61.311 id85 + 0.508 id86 - 1.447 id87 >= 0
 id6:  357.971 id80 + 342.258 id81 + 297.575 id82 + 336.524 id83 + 318.366 id84
       + 317.35 id85 + 280.027 id86 + 333.673 id87 >= 0
 id7:  206.369 id88 + 211.727 id89 + 172.503 id90 + 195.195 id91 + 194.773 id92
       + 194.123 id93 + 152.92 id94 + 169.604 id95 >= 0
 id8:  48.326 id88 + 80.575 id89 + 24.645 id90 + 26.818 id91 + 0.317 id92
       + 35.387 id93 + 6.922 id94 + 11.916 id95 >= 0
 id9:  206.369 id96 + 211.727 id97 + 172.503 id98 + 195.195 id99
       + 194.773 id100 + 194.123 id101 + 152.92 id102 + 169.604 id103 >= 0
 id10: 163.856 id96 + 178.412 id97 + 144.664 id98 + 156.072 id99 + 204.57 id100
       + 186.468 id101 + 177.408 id102 + 156.158 id103 >= 0
 id11: 206.369 id104 + 211.727 id105 + 172.503 id106 + 195.195 id107
       + 194.773 id108 + 194.123 id109 + 152.92 id110 + 169.604 id111 >= 0
 id12: 357.971 id104 + 342.258 id105 + 297.575 id106 + 336.524 id107
       + 318.366 id108 + 317.35 id109 + 280.027 id110 + 333.673 id111 >= 0
 id13: 319.886 id112 + 282.082 id113 + 346.827 id114 + 316.198 id115
```

```
      + 276.324 id116 + 294.06 id117 + 322.572 id118 + 347.483 id119 >= 0
id14: 48.326 id112 + 80.575 id113 + 24.645 id114 + 26.818 id115 + 0.317 id116
      + 35.387 id117 + 6.922 id118 + 11.916 id119 >= 0
id15: 319.886 id120 + 282.082 id121 + 346.827 id122 + 316.198 id123
      + 276.324 id124 + 294.06 id125 + 322.572 id126 + 347.483 id127 >= 0
id16: 163.856 id120 + 178.412 id121 + 144.664 id122 + 156.072 id123
      + 204.57 id124 + 186.468 id125 + 177.408 id126 + 156.158 id127 >= 0
id17: 319.886 id128 + 282.082 id129 + 346.827 id130 + 316.198 id131
      + 276.324 id132 + 294.06 id133 + 322.572 id134 + 347.483 id135 >= 0
id18: 357.971 id128 + 342.258 id129 + 297.575 id130 + 336.524 id131
      + 318.366 id132 + 317.35 id133 + 280.027 id134 + 333.673 id135 >= 0
id19: id64 + id65 + id66 + id67 + id68 + id69 + id70 + id71  = 1
id20: id72 + id73 + id74 + id75 + id76 + id77 + id78 + id79  = 1
id21: id80 + id81 + id82 + id83 + id84 + id85 + id86 + id87  = 1
id22: id88 + id89 + id90 + id91 + id92 + id93 + id94 + id95  = 1
id23: id96 + id97 + id98 + id99 + id100 + id101 + id102 + id103  = 1
id24: id104 + id105 + id106 + id107 + id108 + id109 + id110 + id111  = 1
id25: id112 + id113 + id114 + id115 + id116 + id117 + id118 + id119  = 1
id26: id120 + id121 + id122 + id123 + id124 + id125 + id126 + id127  = 1
id27: id128 + id129 + id130 + id131 + id132 + id133 + id134 + id135  = 1
id28: - 9.267 id64 + 55.74 id65 + 22.346 id66 + 46.326 id67 + 22.239 id68
      + 61.311 id69 + 0.508 id70 - 1.447 id71 + 9.267 id88 - 55.74 id89
      - 22.346 id90 - 46.326 id91 - 22.239 id92 - 61.311 id93 - 0.508 id94
      + 1.447 id95 >= 0
id29: - 9.267 id64 + 55.74 id65 + 22.346 id66 + 46.326 id67 + 22.239 id68
      + 61.311 id69 + 0.508 id70 - 1.447 id71 + 9.267 id112 - 55.74 id113
      - 22.346 id114 - 46.326 id115 - 22.239 id116 - 61.311 id117
      - 0.508 id118 + 1.447 id119 >= 0
id30: - 9.267 id72 + 55.74 id73 + 22.346 id74 + 46.326 id75 + 22.239 id76
      + 61.311 id77 + 0.508 id78 - 1.447 id79 + 9.267 id96 - 55.74 id97
      - 22.346 id98 - 46.326 id99 - 22.239 id100 - 61.311 id101 - 0.508 id102
      + 1.447 id103 >= 0
id31: - 9.267 id72 + 55.74 id73 + 22.346 id74 + 46.326 id75 + 22.239 id76
      + 61.311 id77 + 0.508 id78 - 1.447 id79 + 9.267 id120 - 55.74 id121
      - 22.346 id122 - 46.326 id123 - 22.239 id124 - 61.311 id125
      - 0.508 id126 + 1.447 id127 >= 0
id32: - 9.267 id80 + 55.74 id81 + 22.346 id82 + 46.326 id83 + 22.239 id84
      + 61.311 id85 + 0.508 id86 - 1.447 id87 + 9.267 id104 - 55.74 id105
      - 22.346 id106 - 46.326 id107 - 22.239 id108 - 61.311 id109
      - 0.508 id110 + 1.447 id111 >= 0
id33: - 9.267 id80 + 55.74 id81 + 22.346 id82 + 46.326 id83 + 22.239 id84
      + 61.311 id85 + 0.508 id86 - 1.447 id87 + 9.267 id128 - 55.74 id129
```

```
      - 22.346 id130 - 46.326 id131 - 22.239 id132 - 61.311 id133
      - 0.508 id134 + 1.447 id135 >= 0
id34: - 206.369 id64 - 211.727 id65 - 172.503 id66 - 195.195 id67
      - 194.773 id68 - 194.123 id69 - 152.92 id70 - 169.604 id71
      + 206.369 id88 + 211.727 id89 + 172.503 id90 + 195.195 id91
      + 194.773 id92 + 194.123 id93 + 152.92 id94 + 169.604 id95 >= 0
id35: 206.369 id88 + 211.727 id89 + 172.503 id90 + 195.195 id91 + 194.773 id92
      + 194.123 id93 + 152.92 id94 + 169.604 id95 - 206.369 id112
      - 211.727 id113 - 172.503 id114 - 195.195 id115 - 194.773 id116
      - 194.123 id117 - 152.92 id118 - 169.604 id119 >= 0
id36: - 206.369 id72 - 211.727 id73 - 172.503 id74 - 195.195 id75
      - 194.773 id76 - 194.123 id77 - 152.92 id78 - 169.604 id79
      + 206.369 id96 + 211.727 id97 + 172.503 id98 + 195.195 id99
      + 194.773 id100 + 194.123 id101 + 152.92 id102 + 169.604 id103 >= 0
id37: 206.369 id96 + 211.727 id97 + 172.503 id98 + 195.195 id99
      + 194.773 id100 + 194.123 id101 + 152.92 id102 + 169.604 id103
      - 206.369 id120 - 211.727 id121 - 172.503 id122 - 195.195 id123
      - 194.773 id124 - 194.123 id125 - 152.92 id126 - 169.604 id127 >= 0
id38: - 206.369 id80 - 211.727 id81 - 172.503 id82 - 195.195 id83
      - 194.773 id84 - 194.123 id85 - 152.92 id86 - 169.604 id87
      + 206.369 id104 + 211.727 id105 + 172.503 id106 + 195.195 id107
      + 194.773 id108 + 194.123 id109 + 152.92 id110 + 169.604 id111 >= 0
id39: 206.369 id104 + 211.727 id105 + 172.503 id106 + 195.195 id107
      + 194.773 id108 + 194.123 id109 + 152.92 id110 + 169.604 id111
      - 206.369 id128 - 211.727 id129 - 172.503 id130 - 195.195 id131
      - 194.773 id132 - 194.123 id133 - 152.92 id134 - 169.604 id135 >= 0
id40: - 319.886 id64 - 282.082 id65 - 346.827 id66 - 316.198 id67
      - 276.324 id68 - 294.06 id69 - 322.572 id70 - 347.483 id71
      + 319.886 id112 + 282.082 id113 + 346.827 id114 + 316.198 id115
      + 276.324 id116 + 294.06 id117 + 322.572 id118 + 347.483 id119 >= 0
id41: - 319.886 id88 - 282.082 id89 - 346.827 id90 - 316.198 id91
      - 276.324 id92 - 294.06 id93 - 322.572 id94 - 347.483 id95
      + 319.886 id112 + 282.082 id113 + 346.827 id114 + 316.198 id115
      + 276.324 id116 + 294.06 id117 + 322.572 id118 + 347.483 id119 >= 0
id42: - 319.886 id72 - 282.082 id73 - 346.827 id74 - 316.198 id75
      - 276.324 id76 - 294.06 id77 - 322.572 id78 - 347.483 id79
      + 319.886 id120 + 282.082 id121 + 346.827 id122 + 316.198 id123
      + 276.324 id124 + 294.06 id125 + 322.572 id126 + 347.483 id127 >= 0
id43: - 319.886 id96 - 282.082 id97 - 346.827 id98 - 316.198 id99
      - 276.324 id100 - 294.06 id101 - 322.572 id102 - 347.483 id103
      + 319.886 id120 + 282.082 id121 + 346.827 id122 + 316.198 id123
      + 276.324 id124 + 294.06 id125 + 322.572 id126 + 347.483 id127 >= 0
```

```
id44: - 319.886 id80 - 282.082 id81 - 346.827 id82 - 316.198 id83
      - 276.324 id84 - 294.06 id85 - 322.572 id86 - 347.483 id87
      + 319.886 id128 + 282.082 id129 + 346.827 id130 + 316.198 id131
      + 276.324 id132 + 294.06 id133 + 322.572 id134 + 347.483 id135 >= 0
id45: - 319.886 id104 - 282.082 id105 - 346.827 id106 - 316.198 id107
      - 276.324 id108 - 294.06 id109 - 322.572 id110 - 347.483 id111
      + 319.886 id128 + 282.082 id129 + 346.827 id130 + 316.198 id131
      + 276.324 id132 + 294.06 id133 + 322.572 id134 + 347.483 id135 >= 0
id46: 48.326 id64 + 80.575 id65 + 24.645 id66 + 26.818 id67 + 0.317 id68
      + 35.387 id69 + 6.922 id70 + 11.916 id71 - 48.326 id72 - 80.575 id73
      - 24.645 id74 - 26.818 id75 - 0.317 id76 - 35.387 id77 - 6.922 id78
      - 11.916 id79 >= 0
id47: 48.326 id64 + 80.575 id65 + 24.645 id66 + 26.818 id67 + 0.317 id68
      + 35.387 id69 + 6.922 id70 + 11.916 id71 - 48.326 id80 - 80.575 id81
      - 24.645 id82 - 26.818 id83 - 0.317 id84 - 35.387 id85 - 6.922 id86
      - 11.916 id87 >= 0
id48: - 163.856 id64 - 178.412 id65 - 144.664 id66 - 156.072 id67
      - 204.57 id68 - 186.468 id69 - 177.408 id70 - 156.158 id71
      + 163.856 id72 + 178.412 id73 + 144.664 id74 + 156.072 id75
      + 204.57 id76 + 186.468 id77 + 177.408 id78 + 156.158 id79 >= 0
id49: 163.856 id72 + 178.412 id73 + 144.664 id74 + 156.072 id75 + 204.57 id76
      + 186.468 id77 + 177.408 id78 + 156.158 id79 - 163.856 id80
      - 178.412 id81 - 144.664 id82 - 156.072 id83 - 204.57 id84
      - 186.468 id85 - 177.408 id86 - 156.158 id87 >= 0
id50: - 357.971 id64 - 342.258 id65 - 297.575 id66 - 336.524 id67
      - 318.366 id68 - 317.35 id69 - 280.027 id70 - 333.673 id71
      + 357.971 id80 + 342.258 id81 + 297.575 id82 + 336.524 id83
      + 318.366 id84 + 317.35 id85 + 280.027 id86 + 333.673 id87 >= 0
id51: - 357.971 id72 - 342.258 id73 - 297.575 id74 - 336.524 id75
      - 318.366 id76 - 317.35 id77 - 280.027 id78 - 333.673 id79
      + 357.971 id80 + 342.258 id81 + 297.575 id82 + 336.524 id83
      + 318.366 id84 + 317.35 id85 + 280.027 id86 + 333.673 id87 >= 0
id52: 48.326 id88 + 80.575 id89 + 24.645 id90 + 26.818 id91 + 0.317 id92
      + 35.387 id93 + 6.922 id94 + 11.916 id95 - 48.326 id96 - 80.575 id97
      - 24.645 id98 - 26.818 id99 - 0.317 id100 - 35.387 id101 - 6.922 id102
      - 11.916 id103 >= 0
id53: 48.326 id88 + 80.575 id89 + 24.645 id90 + 26.818 id91 + 0.317 id92
      + 35.387 id93 + 6.922 id94 + 11.916 id95 - 48.326 id104 - 80.575 id105
      - 24.645 id106 - 26.818 id107 - 0.317 id108 - 35.387 id109
      - 6.922 id110 - 11.916 id111 >= 0
id54: - 163.856 id88 - 178.412 id89 - 144.664 id90 - 156.072 id91
      - 204.57 id92 - 186.468 id93 - 177.408 id94 - 156.158 id95
```

```
       + 163.856 id96 + 178.412 id97 + 144.664 id98 + 156.072 id99
       + 204.57 id100 + 186.468 id101 + 177.408 id102 + 156.158 id103 >= 0
 id55: 163.856 id96 + 178.412 id97 + 144.664 id98 + 156.072 id99 + 204.57 id100
       + 186.468 id101 + 177.408 id102 + 156.158 id103 - 163.856 id104
       - 178.412 id105 - 144.664 id106 - 156.072 id107 - 204.57 id108
       - 186.468 id109 - 177.408 id110 - 156.158 id111 >= 0
 id56: - 357.971 id88 - 342.258 id89 - 297.575 id90 - 336.524 id91
       - 318.366 id92 - 317.35 id93 - 280.027 id94 - 333.673 id95
       + 357.971 id104 + 342.258 id105 + 297.575 id106 + 336.524 id107
       + 318.366 id108 + 317.35 id109 + 280.027 id110 + 333.673 id111 >= 0
 id57: - 357.971 id96 - 342.258 id97 - 297.575 id98 - 336.524 id99
       - 318.366 id100 - 317.35 id101 - 280.027 id102 - 333.673 id103
       + 357.971 id104 + 342.258 id105 + 297.575 id106 + 336.524 id107
       + 318.366 id108 + 317.35 id109 + 280.027 id110 + 333.673 id111 >= 0
 id58: 48.326 id112 + 80.575 id113 + 24.645 id114 + 26.818 id115 + 0.317 id116
       + 35.387 id117 + 6.922 id118 + 11.916 id119 - 48.326 id120
       - 80.575 id121 - 24.645 id122 - 26.818 id123 - 0.317 id124
       - 35.387 id125 - 6.922 id126 - 11.916 id127 >= 0
 id59: 48.326 id112 + 80.575 id113 + 24.645 id114 + 26.818 id115 + 0.317 id116
       + 35.387 id117 + 6.922 id118 + 11.916 id119 - 48.326 id128
       - 80.575 id129 - 24.645 id130 - 26.818 id131 - 0.317 id132
       - 35.387 id133 - 6.922 id134 - 11.916 id135 >= 0
 id60: - 163.856 id112 - 178.412 id113 - 144.664 id114 - 156.072 id115
       - 204.57 id116 - 186.468 id117 - 177.408 id118 - 156.158 id119
       + 163.856 id120 + 178.412 id121 + 144.664 id122 + 156.072 id123
       + 204.57 id124 + 186.468 id125 + 177.408 id126 + 156.158 id127 >= 0
 id61: 163.856 id120 + 178.412 id121 + 144.664 id122 + 156.072 id123
       + 204.57 id124 + 186.468 id125 + 177.408 id126 + 156.158 id127
       - 163.856 id128 - 178.412 id129 - 144.664 id130 - 156.072 id131
       - 204.57 id132 - 186.468 id133 - 177.408 id134 - 156.158 id135 >= 0
 id62: - 357.971 id112 - 342.258 id113 - 297.575 id114 - 336.524 id115
       - 318.366 id116 - 317.35 id117 - 280.027 id118 - 333.673 id119
       + 357.971 id128 + 342.258 id129 + 297.575 id130 + 336.524 id131
       + 318.366 id132 + 317.35 id133 + 280.027 id134 + 333.673 id135 >= 0
 id63: - 357.971 id120 - 342.258 id121 - 297.575 id122 - 336.524 id123
       - 318.366 id124 - 317.35 id125 - 280.027 id126 - 333.673 id127
       + 357.971 id128 + 342.258 id129 + 297.575 id130 + 336.524 id131
       + 318.366 id132 + 317.35 id133 + 280.027 id134 + 333.673 id135 >= 0
Bounds
 0 <= id64 <= 1   ...    0 <= id135 <= 1
Generals
 id64  id65  id66  id67  id68  id69  id70  id71  id72  id73  id74  id75  id76
```

```
 id77  id78  id79  id80  id81  id82  id83  id84  id85  id86  id87  id88  id89
 id90  id91  id92  id93  id94  id95  id96  id97  id98  id99  id100  id101
 id102  id103  id104  id105  id106  id107  id108  id109  id110  id111  id112
 id113  id114  id115  id116  id117  id118  id119  id120  id121  id122  id123
 id124  id125  id126  id127  id128  id129  id130  id131  id132  id133  id134
 id135
End
```

Bibliografia

[1] Bistarelli, Rossi, Montanari. *Semiring-based Constraint Solving and Optimization.* Journal of ACM, 44:201-326, 1997.

[2] Stefano Bistarelli. *Soft Constraint Solving and Programming: a general framework.* Tesi di dottorato, Springer Verlag, 2003.

[3] www.cplex.com;

[4] John Nash. *Equilibrium points in n-person games.* Proc. of the National Academy of sciences, 536: 48-49, 1950.

[5] John Nash. *Non cooperative games.* Annals of mathematics, 54: 286-295, 1951.

[6] Tuomas Sandholm. *Automated Mechanism Design: A New Application Area for Search Algorithms.* Proc. CP 2003, Springer Verlag, 2003.

[7] Gibbard. *Manipolation of voting schemes.* Econometrica, 45: 587-602, 1973

[8] Satterwaite. *Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedure and social welfare functions.* Journal of Economic Theory, 10: 187-217, 1975

[9] Vincent Conitzer and Tuomas Sandholm. *Complexity of mechanism design.* In Proceedings of the 18th Annual Conference on Uncertainty in Arti cial Intelligence (UAI-02), pages 103-110, Edmonton, Canada, 2002.

[10] Vincent Conitzer and Tuomas Sandholm. *Automated mechanism design for a selfinterested designer.* In Proceedings of the ACM Conference on

Electronic Commerce (ACM-EC), pages 232-233, San Diego, CA, 2003. Poster paper. Full-length draft available at www.cs.cmu.edu/ sandholm/.

[11] W Vickrey. *Counterspeculation, auctions, and competitive sealed tenders.* Journal of Finance, 16: 8-37, 1961.

[12] Theodore Groves. *Incentives in teams.* Econometrica, 41: 617-631, 1973.

[13] E H Clarke. *Multipart pricing of public goods.* Public Choice, 11:17-33, 1971.

[14] C d'Aspremont and L A Gerard-Varet. *Incentives and incomplete information.* Journal of Public Economics, 11:25-45, 1979.

[15] Kenneth Arrow. *The property rights doctrine and demand revelation under incomplete information.* In M Boskin, editor, Economics and human welfare. New York Academic Press, 1979.

[16] Roger Myerson. *Optimal auction design.* Mathematics of Operation Research, 6: 58-73, 1981.

[17] Eric S Maskin and John Riley. *Optimal multi-unit auctions.* In Frank Hahn, editor, The Economics of Missing Markets, Information, and Games, chapter 14, pages 312-335. Clarendon Press, Oxford, 1989.

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

[19] Vincent Conitzer and Tuomas Sandholm. *Complexity of mechanism design.* In proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02),pages 103-110, Edmonton, Canada, 2002

[20] Vincent Conitzer and Tuomas Sandholm. *Automated Mechanism Design: Complexity results stemming from the single-agent setting.*2002