L'ottimizzazione lagrangiana

Enrico Masala

31 ottobre 2001, rivisto 6 giugno 2006, bugs corretti 16 gen 2009

Sommario

Breve spiegazione sull'ottimizzazione tramite i moltiplicatori di Lagrange e sua applicazione nel caso dell'ottimizzazione discreta di tipo rate-distortion.

1 Introduzione sul rate-distortion

1.1 Curve rate-distortion

Nel caso di compressione *lossy*, esiste un trade-off tra la qualità della codifica (intesa come fedeltà alla sorgente) e la quantità di bit necessari. Questo è esattamente il rate-distortion trade-off, ossia quanto più è alta la distorsione, tanto più basso riesce ad essere il bit-rate, e viceversa.

Si parla quindi di curva rate-distortion, intendendo la variazione della qualità (misurata dalla distorsione rispetto all'originale) al variare del bit-rate di codifica. Queste curve nei casi più semplici sono calcolabili teoricamente, in particolare facendo ipotesi semplificative sulla statistica delle sorgenti. Raramente però queste curve sono utili, sia perché spesso è difficile se non impossibile modellare bene la statistica della sorgente, come nel caso del video e quindi la curva ottenuta non è valida neppure come bound, sia perché i calcoli non dicono come ottenere operativamente una certa codifica.

Si preferisce dunque parlare di *operational rate-distortion*, ossia si sceglie un certo sistema di codifica, che possibilmente catturi in modo efficace le dipendenze statistiche dei dati, e poi si cercano i migliori punti di lavoro per questo specifico sistema. Usando differenti valori per i vari parametri di codifica del sistema (es. Q di quantizzazione nella codifica MPEG), si possono calcolare le corrispondenti distorsioni causate dal sistema di codifica/decodifica, e quindi ottenere un punto della *operational rate-distortion curve*.

1.2 Ottimizzazione R-D

Per prima cosa si deve definire la *coding unit*, sulla quale è possibile modificare i parametri di codifica per ottenere diversi valori di distorsione. Inoltre è necessario definire una *funzione obiettivo* che deve essere per esempio minimizzata, tipicamente la distorsione, e una serie di vincoli da rispettare, per esempio sul numero totale di bit (massimo bit-rate) e/o sul massimo ritardo di trasmissione, perché la soluzione nel caso senza vincoli sarebbe banale.

Per esempio, un tipico problema può essere:
d_n: distorsione coding unit n
r_n: rate coding unit n

$$\min f(d_1, d_2, ..., d_N) \ \text{ con } \ \sum_{i=1}^{N} r_i \leq R_{max}$$

Esempio:

$$\min \sum_{i=1}^{N} d_i \ \text{ con } \ \sum_{i=1}^{N} r_i \leq R_{max}$$

2 Problema caso continuo

2.1 Problema generico

Sistema generico di 2 variabili:

$$\begin{cases} g(x,y) & \text{funzione da minimizzare (o max)} \\ \varphi(x,y) = 0 & \text{vincolo} \end{cases}$$

Soluzione:

$$\begin{cases} \frac{\partial g}{\partial x} = \lambda \frac{\partial \varphi}{\partial x} \\ \frac{\partial g}{\partial y} = \lambda \frac{\partial \varphi}{\partial y} \\ \varphi(x,y) = 0 \end{cases}$$

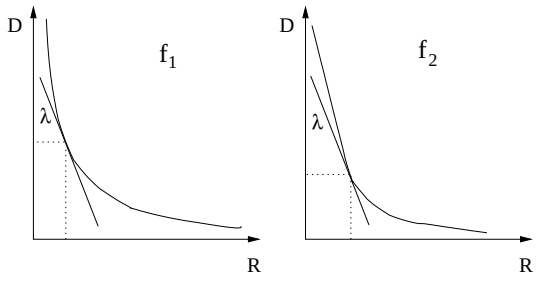

Figura 1: $f_1(R_1)$ e $f_2(R_2)$: Punto di ottimo.

Sistema generico di 3 variabili:

$$\begin{cases} g(x,y,z) & \text{funzione da minimizzare (o max)} \\ \varphi(x,y,z)=0 & \text{vincolo} \end{cases}$$

Soluzione:

$$\begin{cases} \frac{\partial g}{\partial x} = \lambda \frac{\partial \varphi}{\partial x} \\ \frac{\partial g}{\partial y} = \lambda \frac{\partial \varphi}{\partial y} \\ \frac{\partial g}{\partial z} = \lambda \frac{\partial \varphi}{\partial z} \\ \varphi(x,y,z)=0 \end{cases}$$

2.2 Problema rate-distortion

Nel caso del rate distortion, con $D_1 = f_1(R_1)$ e $D_2 = f_2(R_2)$:

$$\begin{cases} g(R_1,R_2) \\ \varphi(R_1,R_2)=0 \end{cases} \Rightarrow \begin{cases} g : f_1(R_1)+f_2(R_2) \\ \varphi : R_1+R_2-r=0 \end{cases}$$

Soluzione:

$$\begin{cases} \frac{\partial f_1}{\partial R_1} = \lambda \frac{\partial \varphi}{\partial R_1} = \lambda \\ \frac{\partial f_2}{\partial R_2} = \lambda \frac{\partial \varphi}{\partial R_2} = \lambda \\ R_1+R_2-r=0 \end{cases}$$

$$\begin{cases} \frac{\partial f_1}{\partial R_1} = \lambda \\ \frac{\partial f_2}{\partial R_2} = \lambda \\ R_1+R_2=r \end{cases}$$

Interpretazione: vedi fig. 1. Avendo le espressioni analitiche di f_1 e f_2, si ricavano R_1 ed R_2 e si sostituiscono nel vincolo, ottenendo così λ.

Muovendosi dal punto di ottimo, in pratica la distorsione totale $D = D_1 + D_2$ può solo aumentare perché se, per esempio, ci si muovesse verso R_2 maggiori, D_2 diminuisce ma D_1 aumenta di più e quindi il totale D aumenta (è necessario infatti mantenere il vincolo $R_1 + R_2 = r$): il punto trovato quindi è il minimo per D.

3 Problema caso discreto

3.1 Soluzione classica

La soluzione classica del problema discreto è basata sul versione discreta dell'ottimizzazione lagrangiana, proposta per la prima volta da Everett nel 1963. L'idea base è la seguente:

Ci sono N coding unit (individuate dall'indice i) e per ognuna di queste ci sono M punti possibili, uno per ogni scelta dei parametri di codifica (individuati dall'indice j). si introduce un moltiplicatore lagrangiano λ e si considera il costo lagrangiano $J_{ij}(\lambda) = d_{ij} + \lambda r_{ij}$.

Il principale risultato afferma che se un certo mapping $j = x(i)$ minimizza

$$\sum_{i=1}^{N} d_{ix(i)} + \lambda \cdot r_{ix(i)}$$

allora questo $x(i)$ è anche la soluzione ottima del problema particolare:

$$\min D(\lambda) = \sum_{i=1}^{N} d_{ij} \quad \text{con} \quad R(\lambda) = \sum_{i=1}^{N} r_i = R_{max}$$

La funzione da minimizzare, per una certa λ data, si può quindi riscrivere come:

$$\min\left(\sum_{i=1}^{N} d_{ix(i)} + \lambda \cdot r_{ix(i)}\right) = \sum_{i=1}^{N} \min(d_{ix(i)} + \lambda \cdot r_{ix(i)})$$

È quindi possibile ottimizzare la scelta dei parametri di codifica delle varie coding unit indipendentemente le une dalle altre (sempre che non vi siano dipendenze, ossia che la scelta di codifica per una non influisca su altre). Per ogni coding unit, il punto sulla caratteristica R-D che minimizza il valore di $d_{ix(i)} + \lambda \cdot r_{ix(i)}$ è quello in cui la tangente di pendenza λ è tangente al convex hull della caratteristica R-D, come si vede nella figura 2.

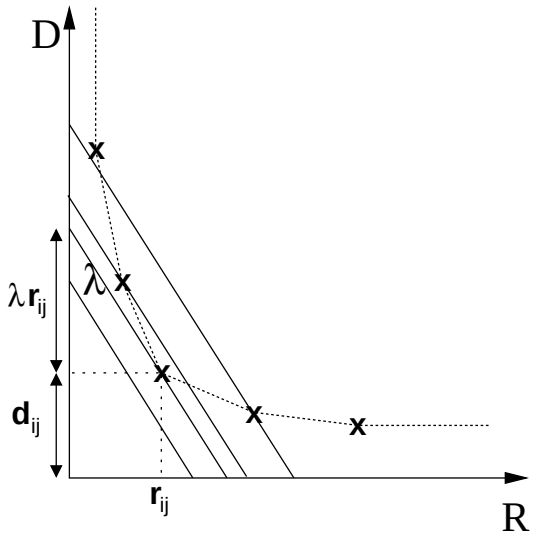

Figura 2: Ottimizzazione per una coding unit.

Il contributo di Everett (il primo che si è occupato di ottimizzazione lagrangiana nel caso discreto) è stato quello di dimostrare che la soluzione trovata con il metodo di lagrange è ottima se esiste una soluzione per cui

$$R(\lambda) = \sum_{i=1}^{N} r_{ix(i)} = R_{max}$$

è esattamente uguale a R_{max}.

Per la ricerca del λ ottimo (il valore per cui $R(\lambda) = R_{max}$) esistono vari algoritmi, tra cui l'algoritmo di bisezione (*bisection algorithm*).

3.1.1 Bisection algorithm

Questo algoritmo consente di individuare il λ ottimo in maniera rapida ed efficiente. Per prima cosa si calcola un valore iniziale di λ, considerando i punti con minimo rate e massimo rate in ogni coding unit. Sommando i rate e le distorsioni minime e massime si ottengono quattro valori, R_l, R_r, D_l, D_r, dove i pedici r e l stanno per 'destra' e 'sinistra'. R_l quindi è la somma dei rate minimi, e D_l è il valore di distorsione corrispondente, ecc. Il valore iniziale di λ si ottiene dalla formula:

$$\lambda = -\frac{D_l - D_r}{R_l - R_r}$$

Utilizzando il valore di λ così calcolato, si ottimizza ogni coding unit in maniera indipendente, ossia per ogni coding unit si calcolano i valori R_i e D_i che si ottengono facendo muovere una retta con pendenza λ verso la caratteristica R-D e vedendo qual è il primo punto che si incontra. A questo punto si ricalcolano i valori totali R_c e D_c dove il pedice c sta per 'current'. Se il valore R_c è superiore al massimo imposto dal vincolo, si ricalcola λ utilizzando $R_r = R_c$ e $D_r = D_c$, altrimenti si utilizza $R_l = R_c$ e $D_l = D_c$. Il procedimento si ferma quando R_c è uguale a R_l (posto che la soluzione per il valore di λ corrente prenda sempre le soluzioni a sinistra (ossia con rate minimo), in caso di indifferenza. A questo punto, i valori R_c e D_c scelti all'ultimo passo per ogni coding unit costituiscono l'allocazione che è il risultato dell'algoritmo.

Nel caso in cui, per l'ultimo valore di λ calcolato, ci sia più di una coding unit soddisfatta all'uguaglianza, significa che piu' di due soluzioni sono allineate sul convex hull su di una retta di pendenza λ. Il caso limite è che tutte le codifiche possibili delle varie coding unit stiano su una retta di pendenza λ. In questo caso l'algoritmo di bisezione non risolve il problema perché la curva R-D diventa una retta. Nel caso in cui piu' di una coding unit è soddisfatta all'uguaglianza, ossia due o più modi di codifica che stanno sul convex hull della coding unit si trovano sulla retta di pendenza λ, è necessario analizzare tutte le combinazioni dei modi di codifica che coinvolgono quelle coding unit in modo da determinare quale sia il punto più vicino al vincolo sulla retta di pendenza λ (nel grafico che comprende tutte le soluzioni).

3.2 Ottimalità della soluzione

L'ottimizzazione lagrangiana ha il problema di non raggiungere sempre la soluzione ottima in senso assoluto, perché non è in grado di raggiungere punti che non stanno sul convex hull della caratteristica R-D. Un esempio di questo caso è illustrato nella figura 3. Avendo come vincolo un budget individuato dalla linea verticale (rate massimo), si vede che il punto B è un punto di ottimo (ha distorsione inferiore rispetto al punto A che sta sul convex hull) ma non viene individuato dalla tecnica di Lagrange.

L'approccio di Lagrange può essere ancora una buona approssimazione quando i punti sul convex hull sono abbastanza densi, per cui la differenza tra la soluzione ottima e la miglior soluzione lagrangiana è piuttosto ridotta. Il vantaggio della

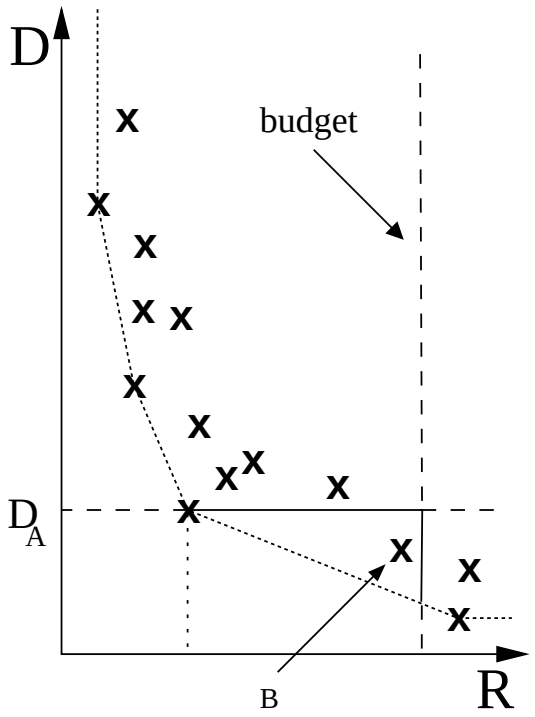

Figura 3: B: Punto non raggiungibile con ottimizzazione lagrangiana.

soluzione lagrangiana è che la complessità dell'algoritmo di soluzione cresce solo linearmente nel numero di coding unit, mentre quella dell'algoritmo di soluzione ottima cresce esponenzialmente.

Per ottenere la soluzione ottima, è necessario riformulare il problema come un problema deterministico di programmazione dinamica. In pratica si costruisce un albero che contiene tutte le possibilità per ogni coding unit, e mano a mano che si procede, si verifica se si finisce in un nodo che è già stato calcolato. In questo caso si mantiene solo il ramo dell'albero la cui distorsione complessiva fino a quel punto è minima, e si procede fino al raggiungimento dei punti per cui è stata decisa una codifica per tutte le coding unit. In pratica al primo livello dell'albero sono rappresentate tutte le possibilità riguardanti la prima coding unit, al secondo livello quelle riguardanti tutte le combinazioni tra la prima e la seconda, e così via.

I vincoli vengono in genere valutati durante la costruzione dell'albero, al fine di minimizzare il numero di calcoli da effettuare. Se per esempio ad un certo punto si arriva ad un nodo che corrisponde ad un valore di rate che eccede il vincolo, tutta la parte di albero che avrebbe dovuto essere costruita a partire da quel nodo viene scartata. In questo modo è possibile tenere facilmente in conto anche altri tipi di vincoli.

Questo tipo di soluzione ha un costo che cresce esponenzialmente con il numero di coding unit, è quindi applicabile solo quando questo numero è ridotto. Lo scarto tra la soluzione ottima e quella ottenuta tramite il metodo dei moltiplicatori di Lagrange è tanto più ridotta quanto più i punti sulla caratteristica R-D sono densi (e quindi si avvicinano ad una curva continua).

3.3 Miglioramento della soluzione

Come già proposto in [2], per migliorare la soluzione e cercare di ovviare al problema della subottimalità, nel caso in cui il vincolo non sia soddisfatto all'uguaglianza, è possibile tentare di migliorare la soluzione con la seguente euristica.

Ordinare le coding units in base alla distorsione. Selezionare i coding modes che danno la maggior riduzione di distorsione, andando nell'ordine prestabilito e posto che ciò sia possibile rimanendo nel vincolo, fino ad esaurire le coding units.

3.4 Pseudocodice dell'algorithmo

```
int solve_units(N, units, lambda) {
 for (i=0; i<N; i++) {
  q=units[i].dist/units[i].rate;
  units[i].eq=0;
  // Fundamental, to avoid
  // rounding errors!!!
  if (lambda - q > EPSILON) {
   units[i].mode=0;
  } else if (lambda - q < -EPSILON ) {
   units[i].mode=1;
  } else {
   // find the solution
   // with minimum rate
   units[i].mode=0;
   units[i].eq=1;
   eq_cnt++;
  }
 }
 return eq_cnt;
}



int lagrange(N, units, Rvinc) {
 Rl=0.0; Dr=0.0;
 Rr=0.0;
 // Compute max rate
 for (i=0; i<N; i++) { Rr+=units[i].rate; }
 if (Rr<=Rvinc) {
  // If constraint is satisfied, return
  for (i=0; i<N; i++) { units[i].mode=1; }
  if (Rr==Rvinc) return;
 }
 Dl=0.0;
 // Compute max distortion
 for (i=0; i<N; i++) { Dl+=units[i].dist; }

 while (1) {
  lambda= -(Dl-Dr)/(Rl-Rr);
  // Solve each coding unit
  // returns the number of coding units
  // for which the choice is indifferent
  eq_cnt=solve_units(N, units, lambda);
  Dc=0.0;
  for (i=0; i<N; i++) {
   if (units[i].mode==0) Dc+=units[i].dist;
  }
  Rc=0.0;
  for (i=0; i<N; i++) {
   if (units[i].mode==1) Rc+=units[i].rate;
  }
  if (Rc < Rvinc) {
   if (Rc==Rl) {
    break;
   } else {
    Rl=Rc; Dl=Dc;
   }
  } else if (Rc > Rvinc) {
   if (Rc==Rr) {
    break;
   } else {
    Rr=Rc; Dr=Dc;
   }
  } else {
   // Constraint is perfectly satisfied
   // This is the optimal solution
   return;
  }
 }
 //Ropt=Rl; Dopt=Dl;
 if (eq_cnt>1) {
  // Test all modes of each coding unit
  // which is satisfied at equality
  test_all_coding_modes(N, units,
             lambda, Rvinc, eq_cnt);
  if (found R == Rvinc) {
   return;
  }
 }
 refine_solution(N, units, Rvinc);
}

// Add new elements as much as the constraint
// allows.  First add elements which cause
// the greatest reduction. Return 1 if the
// constraint is satisfied at equality.
refine_solution(N, units, Rvinc) {
 Dcurr=0.0; Rcurr=0.0;
 cnt=0;
 for (i=0; i<unit_n; i++) {
  if (units[i].mode==0) {
   nontx[cnt]=&(units[i]);
   cnt++;
   Dcurr+=units[i].dist;
  } else {
   Rcurr+=units[i].rate;
  }
```

```
 }
 // sort in decreasing order of dist
 qsort(nontx, cnt);
 for (i=0; i<cnt; i++) {
  if (nontx[i]->rate + Rcurr <= Rvinc) {
   nontx[i]->mode=1;
   Rcurr+=nontx[i]->rate;
   // Subtract from Dcurr because now
   // mode=1, but it was mode=0
   // hence Dcurr was summed
   Dcurr-=nontx[i]->dist;
   if (Rcurr == Rvinc)
    return 1;
  }
 }
 return 0;
}
```

4 Interpretazione articolo TOS

4.1 Slicing fisso (ICME)

Ci sono N coding unit (per esempio possono essere le slice di un frame, oppure i macroblocchi di un frame). Il problema in questo caso diventa della forma:

$$\begin{cases} g(R_1, ..., R_n) = \sum_{i=1}^{N} f_i(R_i) \\ \varphi(R_1, ..., R_n) = \sum_{i=1}^{N} R_i \leq R_{max} \end{cases}$$

Per semplificare la risoluzione, ipotizziamo che tutte le coding unit abbiano rate costante $R_i = R$. Inoltre ipotizziamo che la funzione da minimizzare sia la distorsione totale, in uno scenario di rete diffserv, e la rete abbia solo due tipologie di traffico: premium e best effort; se la coding unit è inviata utilizzando la prima tipologia, la distorsione prodotta è 0, altrimenti la distorsione potenziale producibile è pari a D_i, variabile secondo la coding unit i. Supponiamo che questa distorsione si verifichi sempre, per semplicità. (Qui in realtà si dovrebbe utilizzare la distorsione attesa, quindi moltiplicare per la probabilità). Inoltre è presente un vincolo sul massimo rate della tipologia premium, pari a R_{pr}. La formulazione del problema diventa la seguente:

$$f_i(R_i) = \begin{cases} 0 & \text{se } R_i = R \\ D_i & \text{se } R_i = 0 \end{cases}$$

Per comodità introduciamo una variabile $x(i)$ che assume valore 0 quando $R_i = 0$ e valore 1 quando $R_i = R$.

$$x(i) = \begin{cases} 1 & \text{se } R_i = R \text{ (inviata premium)} \\ 0 & \text{se } R_i = 0 \text{ (inviata best effort)} \end{cases}$$

$$f_i(x(i)) = \begin{cases} D_i & \text{se } x(i) = 0 \\ 0 & \text{se } x(i) = 1 \end{cases}$$

Il problema assume la forma:

$$\begin{cases} \min \sum_{i=1}^{N} f_i(x(i)) \\ R \sum_{i=1}^{N} x(i) \leq R_{pr} \end{cases}$$

$$\begin{cases} \min \sum_{i=1}^{N} D_i(1 - x(i)) \\ R \sum_{i=1}^{N} x(i) \leq R_{pr} \end{cases}$$

$$\begin{cases} \min \sum_{x(i)=0} D_i \\ \sum_{x=1}^{N} x(i) \leq \rho \end{cases}$$

con $\rho = \frac{R_{max}}{R}$.

La soluzione intuitiva si ottiene partendo con $x(i) = 0 \ \forall i$, scegliendo le coding unit con distorsione maggiore e ponendo per queste $x(i) = 1$ (in modo che la riduzione di distorsione sia la massima possibile) fino a raggiungere il vincolo $\sum_{x=1}^{N} x(i) \leq \rho$ in modo che $\sum_{x(i)\neq 0}^{N} D_i$ sia calcolata con le D_i più piccole possibili.

Volendo risolvere con i moltiplicatori di Lagrange, si tiene in conto la quantità $J_i = D_i + \lambda R_i$, per ogni coding unit, e si procede all'applicazione dell'algoritmo di bisezione. Prima di fare questo, ricaviamo la soluzione del problema per una singola coding unit nel caso in cui sia fissato λ (figura 4).

Considerando la retta di pendenza λ, e i due punti sul piano R-D di coordinate $(R, 0)$ e $(0, D_i)$, la retta, muovendosi da sinistra verso destra, incontra prima il punto $(0, D_i)$ se $D_i \leq \lambda R$, oppure il punto $(R, 0)$ se $D_i \geq \lambda R$ (avendo supposto $R_i = R$ per tutte le conding unit). Il caso in cui $D_i = \lambda R_i$ è da gestire in modo accurato: per esempio, selezionare sempre come soluzione il punto "a sinistra", ossia $(0, D_i)$, in modo da gestire i casi particolari per cui c'è uguaglianza per più di una coding unit.

Applichiamo ora l'algoritmo di bisezione:

$$\lambda_{iniz} = -\frac{D_l - D_r}{R_l - R_r} = -\frac{D_l - 0}{0 - R_r}$$

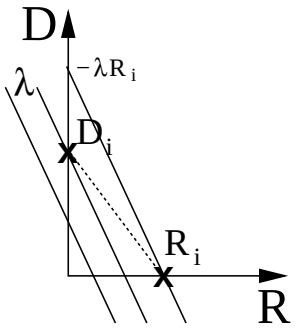

Figura 4: Ottimizzazione coding unit i.

$$\lambda_{iniz} = \frac{\sum_{x=1}^{N} D_i}{\sum_{x=1}^{N} R_i} = \frac{\sum_{x=1}^{N} D_i}{NR}$$

Si considera quante coding unit presentano $D_i \leq \lambda R$ e per queste si pone $x(i) = 0$, e $x(i) = 1$ nell'altro caso. Si calcola $D_c = \sum_{i=0}^{N} D_{ix(i)} = \sum x(i) = 0 D_i$ (essendo $D_{ix(i)} = 0$ se $x(i) = 1$ (vedi grafico 4). Il rate corrispondente è dato da $R_{curr} = \sum_{i=0}^{N} R_{ix(i)} = \sum x(i) = 1R$. Se $R_c > R_{pr}$ allora $R_r = R_c$ e $D_r = D_c$, altrimenti $R_l = R_c$ e $D_l = D_c$. Se si ha $R_c = R_{pr}$ allora l'algoritmo termina immediatamente, altrimenti termina quando si verifica la condizione $R_c = R_l$. In entrambi i casi si ha un'allocazione $\bar{x}$ dove gli $x(i) = 1$ indicano che la coding unit deve essere codificata come nel punto $(R, 0)$ quindi *premium*, altrimenti come nel punto $(0, D_i)$ ossia *best effort*.

Ragionando sul diagramma R-D contenente tutte le coding unit, si vede che il risultato deve necessariamente essere un punto appartenente al convex hull della caratteristica R-D. La caratteristica si ottiene disegnando sul piano i punti relativi a tutte le combinazioni che può assumere il vettore $\bar{x}$ (vedi figura 5).

I punti che formano il convex hull sono quelli con ordinata (distorsione) minore fissato un certo rate. La combinazione che ha la minore distorsione fissato un certo rate è quella che si ottiene a partire dal rate 0 e distorsione massima, sottraendo via via (ossia ponendo $x(i) = 1$) le coding unit con D_i il più elevata possibile. Poichè la soluzione è uno di questi punti, come è noto dal metodo di Lagrange, è sufficiente individuare quale punto è più vicino al rate massimo ammesso R_{pr}. Per fare ciò si può utilizzare un algoritmo semplificato come

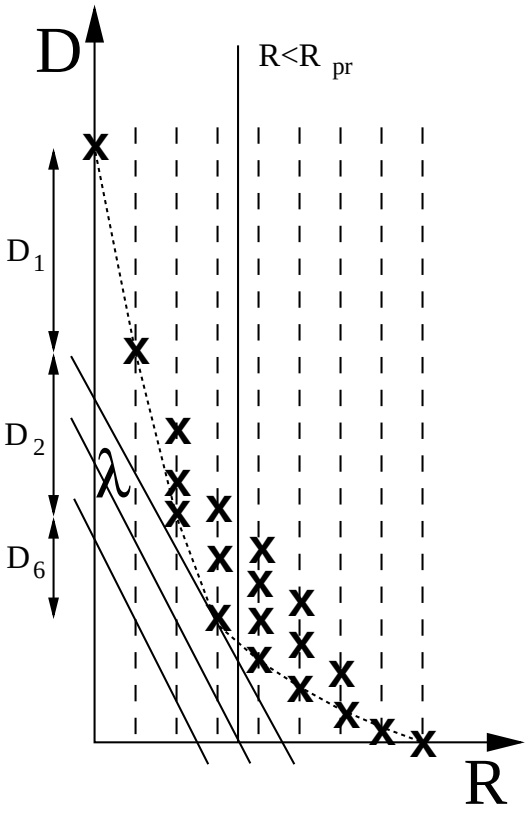

Figura 5: Caratteristica R-D complessiva, vedi anche esempio.

quello descritto, che consiste nel marcare via via le coding unit con distorsione D_i maggiore rimaste non marcate, fino al raggiungimento per difetto del vincolo sul massimo rate R_{pr}.

Esempio:
Coding unit: $D_i = 10, 8, 2, 5, 3, 7, 6$
Rate: $R_i = R = 1 \ \ \forall i$
Banda premium: $R_{pr} = 3.1R = 3.1$

$D_l = 41, R_l = 0$
$D_r = 0, R_r = 7$
$\lambda = -\frac{41-0}{0-7} = \frac{41}{7} = 5.85$
$x(i) = 1, 1, 0, 0, 0, 1, 1$
(perché $10 > 5.85 \rightarrow 1$; $\quad 8 > 5.85 \rightarrow 1$; etc.)
$D_c = 2 + 5 + 3 = 10$
$R_c = 4R > 3.1R \Rightarrow R_r = 4, D_r = 10$

$D_l = 41, R_l = 0$
$D_r = 10, R_r = 4$
$\lambda = -\frac{41-10}{0-4} = \frac{31}{4} = 7.75$
$x(i) = 1, 1, 0, 0, 0, 0, 0$
$D_c = 2 + 5 + 3 + 7 + 6 = 23$
$R_c = 2R < 3.1R \Rightarrow R_l = 2, D_l = 23$

$D_l = 23, R_l = 2$
$D_r = 10, R_r = 4$
$\lambda = -\frac{23-10}{2-4} = \frac{13}{2} = 6.5$
$x(i) = 1, 1, 0, 0, 0, 1, 0$
$D_c = 2 + 5 + 3 + 6 = 16$
$R_c = 3R < 3.1R \Rightarrow R_l = 3, D_l = 16$

$D_l = 16, R_l = 3$
$D_r = 10, R_r = 4$
$\lambda = -\frac{16-10}{3-4} = \frac{6}{1} = 6$
$x(i) = 1, 1, 0, 0, 0, 1, 0$
(perché $6 \leq 6 \rightarrow 0$ (scelgo la soluz. a sinistra))
$D_c = 2 + 5 + 3 + 6 = 16$
$R_c = 3R = R_l \Rightarrow$ fine algoritmo

Marcatura finale:
$x(i) = 1, 1, 0, 0, 0, 1, 0$
Coding unit *premium*: $i = 1, 2, 6$ corrispondenti alle distorsioni maggiori: $D_1 = 10, D_2 = 8, D_6 = 7$.

5 Appendice

5.1 Dimostrazione metodo di Lagrange

Dimostrazione del metodo dei moltiplicatori di Lagrange nel caso di funzione con due variabili e un solo vincolo.

$$\begin{cases} g(x,y) & \text{funzione da minimizzare (o max)} \\ \varphi(x,y) = 0 & \text{vincolo} \end{cases}$$

$X = (x, y) \in \mathbf{R}^2$

In X_0 punto di minimo deve essere:

$$\nabla g(X_0) = 0 \ \text{ e } \ \nabla^2 g(X_0) \geq 0$$

$$\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right) = \overline{0}$$

$$\frac{\partial g}{\partial x} = 0, \frac{\partial g}{\partial y} = 0$$

$$\frac{\partial g}{\partial x} + \frac{\partial g}{\partial x} = 0$$

Per la regola di derivazione in catena:

$$\frac{\partial g}{\partial x} 1 + \frac{\partial g}{\partial y}\frac{\partial y}{\partial x} = 0$$

$$\frac{\partial g}{\partial x} 1 + \frac{\partial g}{\partial y} f'(X_0) = 0$$

avendo indicato l'equazione in forma esplicita del vincolo con f. Proseguendo si ottiene

$$\overline{\nabla} g(X_0) \cdot (1, f'(X_0)) = 0$$

cioè il gradiente $\overline{\nabla} g$ deve essere $\perp$ al vettore tangente alla curva $y = f(x)$ in X_0: $\Big(1, f'(X_0)\Big)$ identifica il vettore tangente alla curva $y = f(x)$ in X_0.

Si consideri $\varphi(x, y) = 0$: essa è localmente esplicitabile rispetto a y se $\frac{\partial \varphi}{\partial y}(X_0) \neq 0$.

Si ha che:

$$f'(x) = -\frac{\partial \varphi}{\partial x} \Big/ \frac{\partial \varphi}{\partial y}$$

$$\Big(1, f'(x)\Big) = \Big(1, -\frac{\partial \varphi}{\partial x} \Big/ \frac{\partial \varphi}{\partial y}\Big) \parallel \left(-\frac{\partial \varphi}{\partial y}, \frac{\partial \varphi}{\partial x}\right)$$

$$\left(-\frac{\partial\varphi}{\partial y}, \frac{\partial\varphi}{\partial x}\right) \cdot \left(\frac{\partial\varphi}{\partial x}, \frac{\partial\varphi}{\partial y}\right) = 0$$

$$\Big(1, f'(x)\Big) \perp \left(\frac{\partial\varphi}{\partial x}, \frac{\partial\varphi}{\partial y}\right) = \overline{\nabla}\varphi$$

Quindi, poichè:

$$\Big(1, f'(x)\Big) \perp \overline{\nabla}\varphi$$

e

$$\Big(1, f'(x)\Big) \perp \overline{\nabla} g$$

allora

$$\overline{\nabla} g \parallel \overline{\nabla}\varphi$$

per cui

$$\begin{cases} \frac{\partial g}{\partial x} = \lambda \frac{\partial\varphi}{\partial x} \\ \frac{\partial g}{\partial y} = \lambda \frac{\partial\varphi}{\partial y} \end{cases}$$

6 Riferimenti

Molte informazioni utili ed approfondimenti si possono trovare in [1] (prima applicazione dell'ottimizzazione lagrangiana nel caso discreto), [2] (applicazione a un set di quantizzatori), [3] (applicazione alle wavelet), [4] (spiega bene l'algoritmo di bisezione), [5] (survey sull'argomento dell'ottimizzazione R-D, è stato molto utile se non fondamentale per scrivere questo documento), [6] (applicazione dell'RD al caso della codifica video, in particolare la scelta MB Intra, Inter, Skipped).

Riferimenti bibliografici

[1] H. Everett, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research*, vol. 11, pp. 399–417, 1963.

[2] A. Gersho Y. Shoham, "Efficient Bit Allocation for an Arbitrary Set of Quantizers," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 9, pp. 1445–1453, September 1988.

[3] M. Vetterli K. Ramchandran, "Best Wavelet Packet Bases in a Rate-Distortion Sense," *IEEE Transactions on Image Processing*, vol. 2, no. 2, pp. 160–175, April 1993.

[4] D. L. Jones B. S. Krongold, K. Ramchandran, "Computationally Efficient Optimal Power Allocation Algorithm for Multicarrier Communication Systems," in *Int. Conf. Communications*, June 1998.

[5] K. Ramchandran A. Ortega, "Rate-distortion Methods for Image and Video Compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, November 1998.

[6] T. Wiegand G.J. Sullivan, "Rate-distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, November 1998.