Parte 2: variazione entropia al variare della dimensione della "base statistica"

Esperimenti sulla stima entropica

Di *leonardo maffi*

Versione 0.5 del 18 Gennaio 2003.

[Scarica il software in Delphi 5 e i dati risultanti (zippati).]

Contenuti:
Qui mostro e discuto delle misurazioni su come muta localmente l'entropia di alcuni file, e al contempo come si comportano alcuni compressori dati man mano che accumulano informazioni (statistiche) sul file che vanno comprimendo. Ho eseguito queste misurazioni perché non sono riuscito a trovare niente di simile altrove, e l'argomento mi pareva interessante.

Nota per la stampa: questo documento contiene grafici nei quali i colori sono usati per denotare dati diversi. Stampandolo in B/N è possibile che parte di tali informazioni vengano perse. Scusate.

Dati su cui sono state fatte le misurazioni:
Ho scelto due file ben noti e facilmente reperibili, in modo da rendere queste misurazioni più ripetibili, per cui ho usato due file tratti dalla suite di test ACT:
http://compression.ca/act-files.html
- Il file testuale è la traduzione inglese dei Tre Moschettieri, di Alexandre Dumas, (490 KB zippati, 1'344'379 byte non zippati):
ftp://sunsite.unc.edu/pub/docs/books/gutenberg/etext98/1musk10.zip
- Un file eseguibile su Windows, Netscape Navigator v4.06 (1352 KB zippato, 2'934'336 byte non zippati):
http://compression.ca/files/act2-netscape406.zip

Compressori utilizzati:
Per fare i test di compressione e stima entropica ho utilizzato alcuni dei migliori programmi disponibili, dato che i compressori migliori si avvicinano di più alla "vera entropia", anche se per alcuni motivi non sempre i migliori in assoluto (nota: in questo documento non mi riferisco all'entropia di Shannon di ennesimo ordine, ma a quella calcolata sul contesto, cioè ai valori di entropia che si ottengono comprimendo un file con software che lavorano su contesti a lunghezza più o meno variabile. Di solito su file normali i valori dell'entropia contestuale sono significativamente minori dell'entropia di Shannon di ordine 0, 1, 2 o anche più.).
Per fare queste misurazioni ho dovuto fare moltissime prove di compressione, automatizzate attraverso dei programmini Delphi, per cui ho utilizzato compressori a linea di comando (per questo ho utilizzato Compressia V. 0.98 e non 0.99b, dato che quest'ultimo non funziona a linea di comando. Vista la quantità di misurazioni che dovevo fare (cioè di compressioni dati) ho dovuto scegliere dei compressori molto veloci. Fortunatamente alcuni dei compressori migliori sono anche molto veloci (ad esempio PPMnostr).
Molti di questi compressori possono essere trovati qui:
http://ftp.unicamp.br/pub/pc/archivers/
http://ftp.vse.cz/msdos/SAC/pc/pack/
Compressori utilizzati:
- *PKZIP* V. shareware 2.04g del 02-01-93.
- *Compressia* 0.98 beta, è basato sull'algoritmo BWT (trasformazione a blocchi con ordinamento), e sfrutta una serie di filtri testuali (la versione 0.99b con interfaccia grafica ha anche un filtro specifico per testi in Inglese).
http://www.compressia.com/
- *PPMonstr* V. i1, di Dmitry Shkarin, compressore PPMII ottimizzato per la velocità. Una variante è utilizzata anche nel 7-Zip, nel WinRar3+, e in Entropy (attualmente il miglior compressore testuale)
http://ftp.unicamp.br/pub/pc/archivers/ppmdi1.rar
- *Rar 3* a linea di comando, basato su una variante di PPMmonstr, ma grazie all'applicazione automatica di vari filtri e vari algoritmi, è più stabile al variare del tipo di file.

http://www.rarlab.com/
- *DC*, Distance Coder v0.98beta (c) 1999-2000 Edgar Binder.
Un compressore basato sulla BWT, ma che usa un modello del tutto diverso dalla MTF. Per funzionare l'ho dovuto usare con l'opzione -n per disabilitare l'individuazione del processore.
http://ftp.unicamp.br/pub/pc/archivers/dc124.zip
(Questo compressore non è stato utilizzato per fare grafici di questo articolo, ma è stato usato per fare dei test).

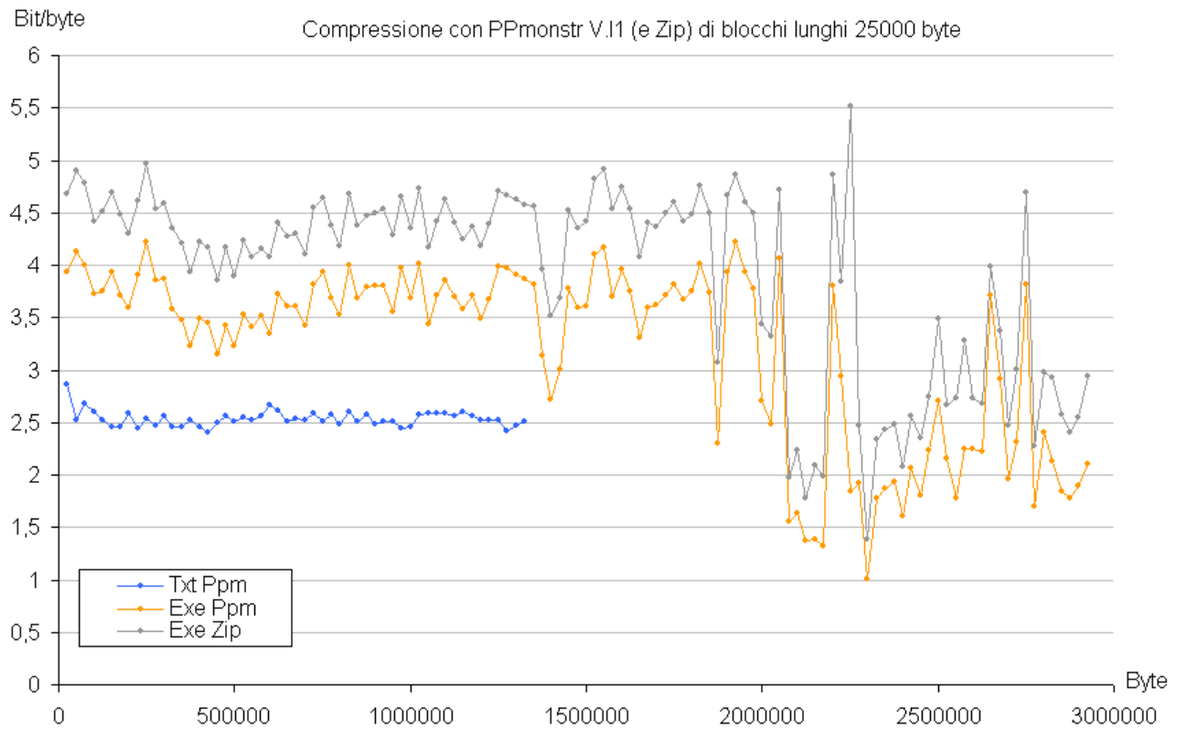
Parametri utilizzati per i compressori:
Dopo vari tentativi questi sono stati i parametri che ho utilizzato, che producono i file compressi minimi:
Rar a -m5 -md4096
PPMonstr e -m170 -o32 -f
Dc e -n -b4096
(Compressia) compcl c -b5

Altri software utilizzati:
- I piccoli programmini di analisi e di elaborazione sono scritti in *Delphi 5*, e sono linkati in questa pagina. Non sono difficili da tradurre in un altro linguaggio, ad esempio C.
- I grafici riassuntivi sono stati fatti con un vecchia versione di *Excel*.
- *Upct*, Ultra Precision Command Timer 1.6 - Freeware (C) 1993 di Erik de Neve, per la misurazione precisa dei tempi di compressione (la versione funzionante su Win2000 occupa 7 KB)
ftp://ftp.sac.sk/pub/sac/utilmisc/upct16.zip
- Xdelta V. 1.1.3 (per Windows). Ottimo Software che estrae differenze tra un primo e un secondo file, e le comprime in piccole patch che possono essere applicate al primo file per ottenere il secondo file.
http://www.eng.uwaterloo.ca/~ejones/software/xdelta-win32.html
Linux: http://sourceforge.net/projects/xdelta/

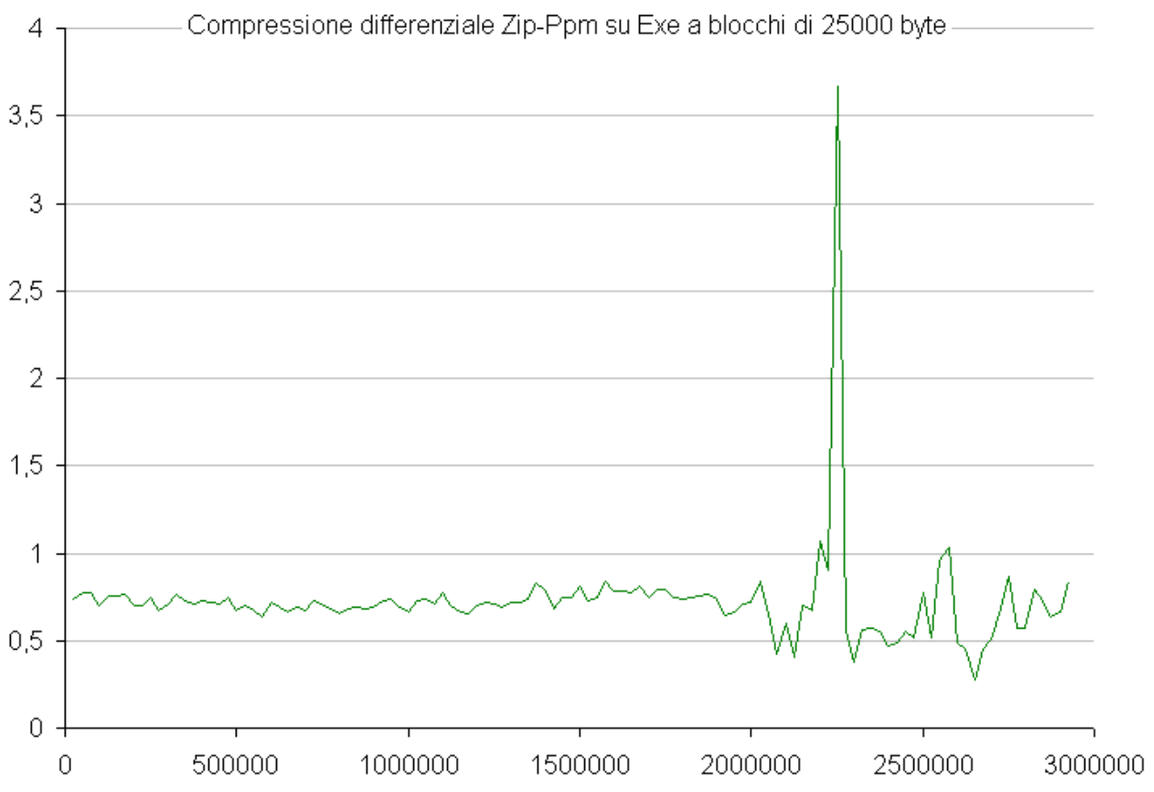
Parte 1: misurazioni dell'entropia locale

Nel primo gruppo di test divido il file-test (il file in ingresso, cioè il testo dei tre moschettieri oppure il Netscape) in piccoli blocchi (adiacenti e non sovrapposti) e comprimo ciascuno indipendentemente dall'altro. In questo modo posso misurare come varia localmente l'entropia (quella basata sul contesto) all'interno del file. Facendo un grafico di questi valori si può fare un'analisi al file, si possono scoprire zone a bassa entropia, ad esempio zone di testo o bitmap uniformi, si possono scoprire discontinuità nel grafico che denotano ad esempio un punto di giunzione tra due file, eccetera.
La lunghezza del blocco d'analisi non è un parametro critico, ma occorre trovare un compromesso tra vari fattori. Dei blocchi corti permettono una maggiore risoluzione "spaziale", cioè permettono di osservare meglio come varia l'entropia nel file, ma hanno anche l'effetto di aumentare il rumore di misurazione, cioè i alori entropici risultanti sono localmente mutevoli e suscettibili a piccole variazioni locali non molto significative.
Comunque un grafico entropico ottenuto con blocchi grandi non equivale ad una versione "smussata" di un grafico ottenuto con blocchi molto piccoli, dato che all'aumentare della lunghezza dei blocchi i compressori riescono ad ottenere una base statistica più ampia, e quindi a dare dei valori di entropia più bassi. Alla fine mi sono assestato su blocchi lunghi 25'000 byte (comunque anche blocchi da 5'000 a 50'000 byte danno risultati qualitativamente simili).

Questo grafico riassume gran parte dei risultati:

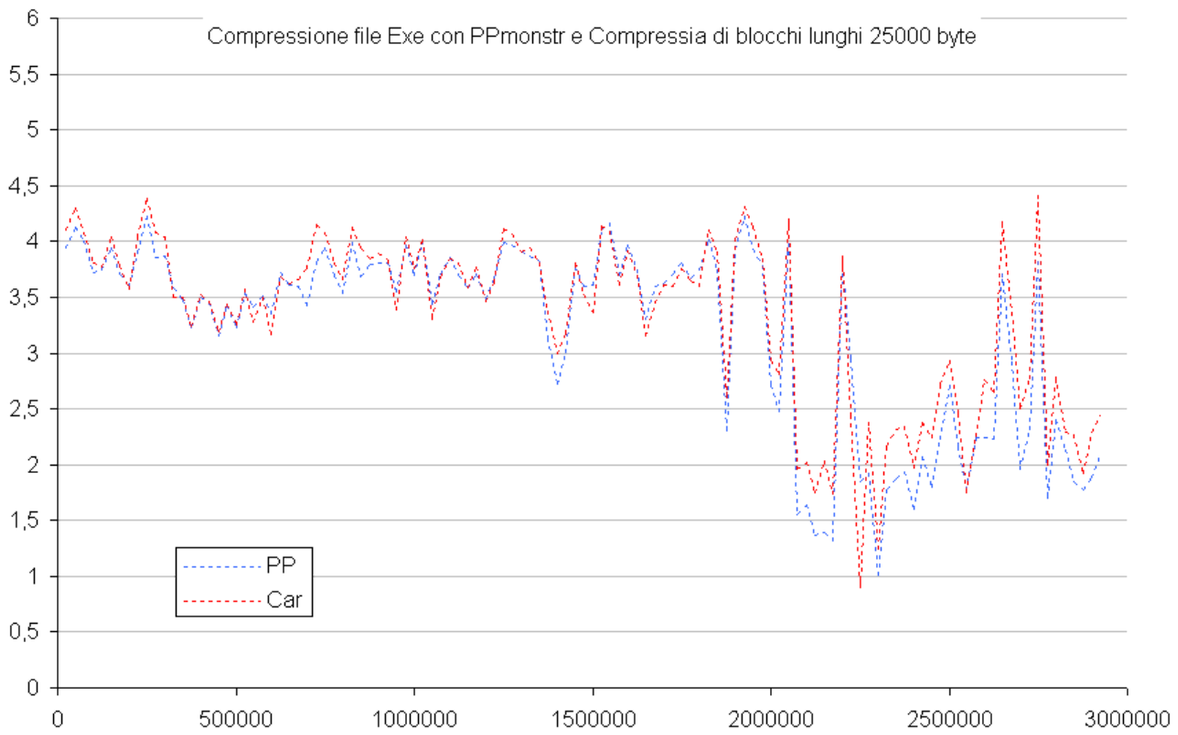


Come si vede il file Txt (testuale dei Tre Moschettieri) è piuttosto uniforme, il PPmonstr ha compresso sui pezzetti lunghi 25000 byte a circa 2.5 bpc (bit/carattere o bit/byte).
Invece i blocchetti di 25000 byte dell'Exe (Netscape) rivelano una prima parte compressa con PPmonstr a circa 4 bpc, seguita da una lunga zona a circa 3.5 e infine una zona molto variabile che viene compressa più di un file testuale, a circa 2-2.3 bps.
Per stimare l'affidabilità del Ppm nello stimare l'entropia, ho compresso il file Exe con PKZip, ottenendo come prevedibile tassi di compressione mediamente minori, ma comunque notevolmente paralleli.
Questo può essere visto meglio nel seguente grafico, che mostra la differenza tra la compressione (sul file Exe) data dallo zip e quella data dal PPmonstr:

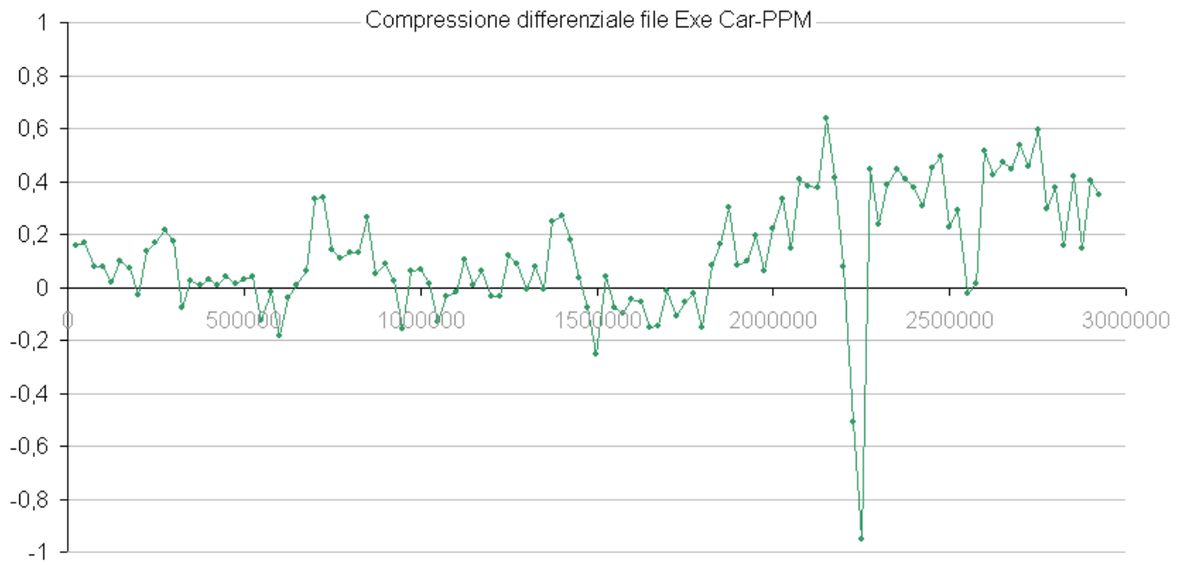


La differenza è sempre positiva (cioè il PPmonstr è sempre migliore dello Zip) ed è notevolmente costante intorno a 0.7 bps. In un punto si nota un picco nel quale lo Zip pare essere molto incapace di comprimere rispetto al PPm.

Sempre sul file Exe diviso a blocchi di 25000 byte ho effettuato un cronfronto tra PPm e Compressia 0.98 (Car):



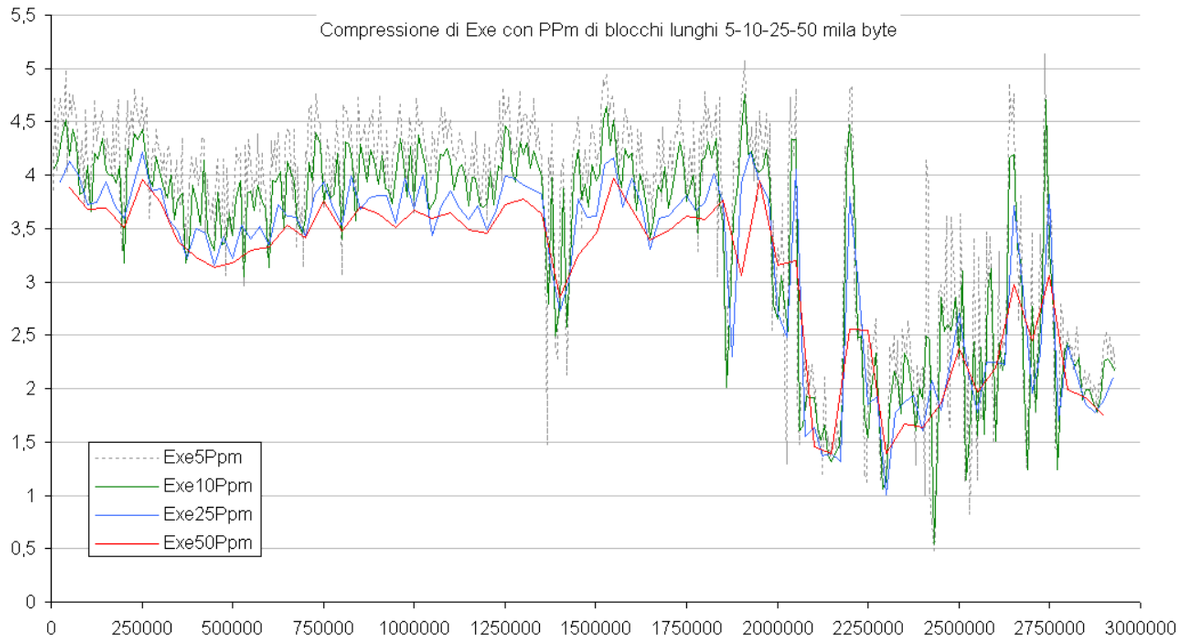
Tenendo di conto che i due programmi utilizzano algornitmi del tutto diversi (e a loro volta del tutto diversi da quello dello Zip) i valori sono sempre vicinissimi. Anche in questo caso ho effettuato il grafico differenza (Car-PPm):



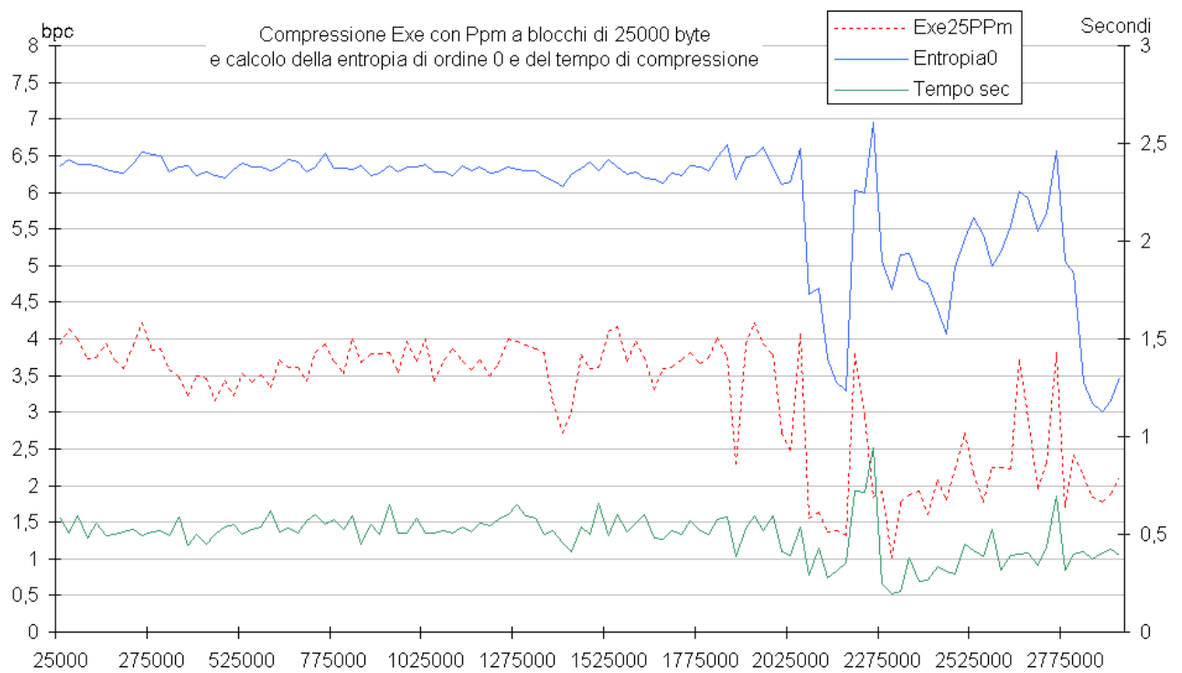
Qui i valori non sono sempre positivi dato che talvonta il PPm risulta migliore del Car.
Nel complesso posso dire che l'uso di Compressia o PPmonstr sembra dare risultati affidabili nella misurazione dell'entropia contestuale locale dei file. Perfino lo zip parrebbe essere sufficiente, ma spesso non è necessario dato che questi programmi sono disponibili gratuitamente (in particolare PPmonstr, mentre Compressia probabilmente diventarà commerciale), e sono veloci in maniera quasi paragonabile allo zip.

Si potrebbe realizzare un piccolo programmino di analisi che dato un file qualunque ne mostra il grafico di entropia locale. Per calcolare il grafico in fretta credo sarebbe meglio utilizzare una libreria integrata di compressione con programma che effettua il grafico. I sorgenti del PPmonstr sono disponibli, oppure si può utilizzare le librerie del Bzip2, che forse sono più veloci e quasi altrettanto precise nella stima dell'entropia.

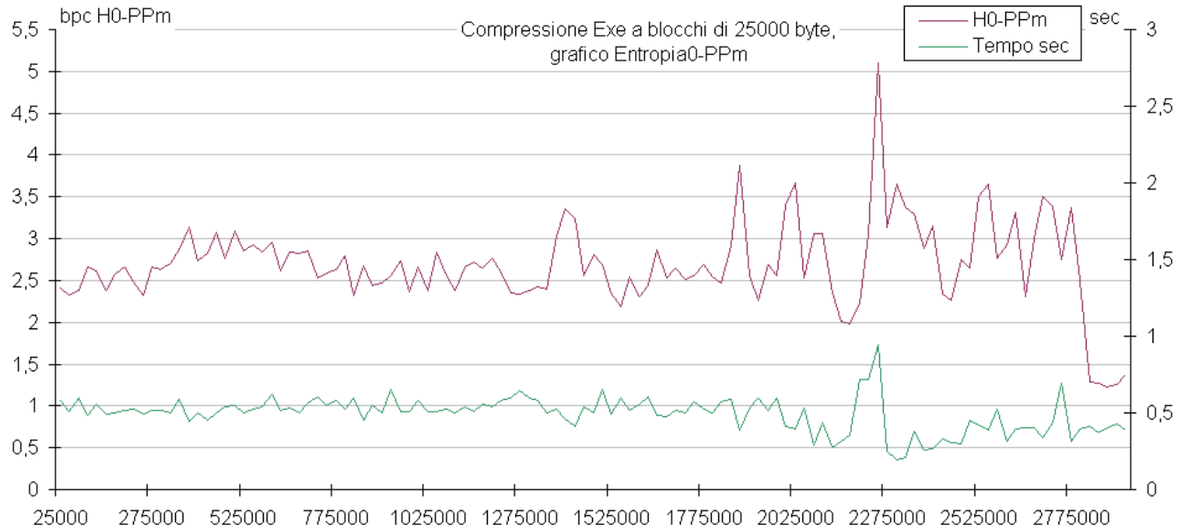
Ho messo a confronto i grafici ottenuti comprimendo con PPmonstr il Netscape diviso a blocchetti di varie dimensioni (5000, 10000, 25000 e 50000 byte). In questo grafico si può vedere che i risultati sono analoghi, anche se all'aumentare della lunghezza dei blocchietti il compressore riesce ad ottenere un bpc via via minore:



Quale è l'efficienza di un programma come PPmonstr nel comprimere? Quanto si avvicina all'entropia di Shannon? Nel seguente grafico si può osservare come l'entropia contestuale del PPmonstr risulti sempre decisamente minore dell'entropia H0 di Shannon di ordine zero (calcolata per ogni blocco con un istrogramma delle frequenze dei singoli byte). Questo mostra che il contesto usato da PPmonstr è sempre decisamente più lungo di un singolo carattere.
Nel grafico si può osservare anche i valori del tempo di compressione di ogni singolo blocchetto di 25000 byte. Come si vede non è costante, i tempi medi sono intorno al mezzo secondo (asse graduato sulla destra), ma ci sono picchi di quasi un secondo. La misurazione dei tempi è stata effettuata una sola volta.



Per poter osservare meglio la relazione tra tasso di compressione e tempo di compressione ho realizzato un'altro grafico. In esso confronto i tempi di compressione con la differenza tra l'entropia di ordine zero e la compressione del PPmonstr (cioè in un certo senso l'utilità dei lunghi contesti usati da PPmonstr). Nel grafico si vede che alcuni picchi sono concidenti, ma le coincidenze non mi paiono significative.



Lo scopo di questo grafico sarebbe stato indagare se in un compressore come PPmonstr si può parlare di "transizioni di fase" computazionale, cioè un aumento notevole di tempo di eseguzione in casi di transizione tra problemi banali e problemi impossibili (cioè tra file molto facili da comprimere e file incomprimibili [1]), ma i risultati per quanto interessanti non suggeriscono alcuna conclusione. Durante la compressione PPmonstr mostra anche la RAM utilizzata per la compressione. Di solito a parità di lunghezza del file da comprimere tale programma (basato su un algorigmo PPMII, simile al PPM) utilizza più Ram per file con un'entropia contestuale maggiore.

Ho effettuato un secondo gruppo di test per misurare quanto la dimensione della "base statistica" influenza i compressori. Man mano che un compressore elabora un lungo file, raccoglie "statistiche" (inteso in senso generalizzato, dato che ad esempio gli algoritmi come la BWT non computano realmente delle statistiche di nessun tipo) sul file stesso, e le usa per riuscire a prevedere meglio il contenuto successivo del file, e quindi a comprimerlo con ad un tasso bps minore. Questo ovviamente è vero solo se il file è uniforme (più o meno stazionario), cioè contiene solo dati dello stesso tipo (questo è vero per il nostro file Txt, ma non per Exe).
Per farlo ho estratto i soli primi 1000 byte da un file di prova, l'ho compresso con un compressore e ho memorizato la dimensione del file risultante. Poi ho preso i primi 2000 byte del file di prova (inclusi i 1000 precedenti) e ho compresso anche essi. E così via.
Man mano che proseguivo ho allungato la dimensione del blocco inziale, in modo da fare un mumero di compressioni non eccessivo. Per una dimensione inferiore a 20'000 byte ho incrementato di 1000 byte, per una dimensione compresa tra 20'000 e 200'000 ho incrementato di 2000 byte e così via:

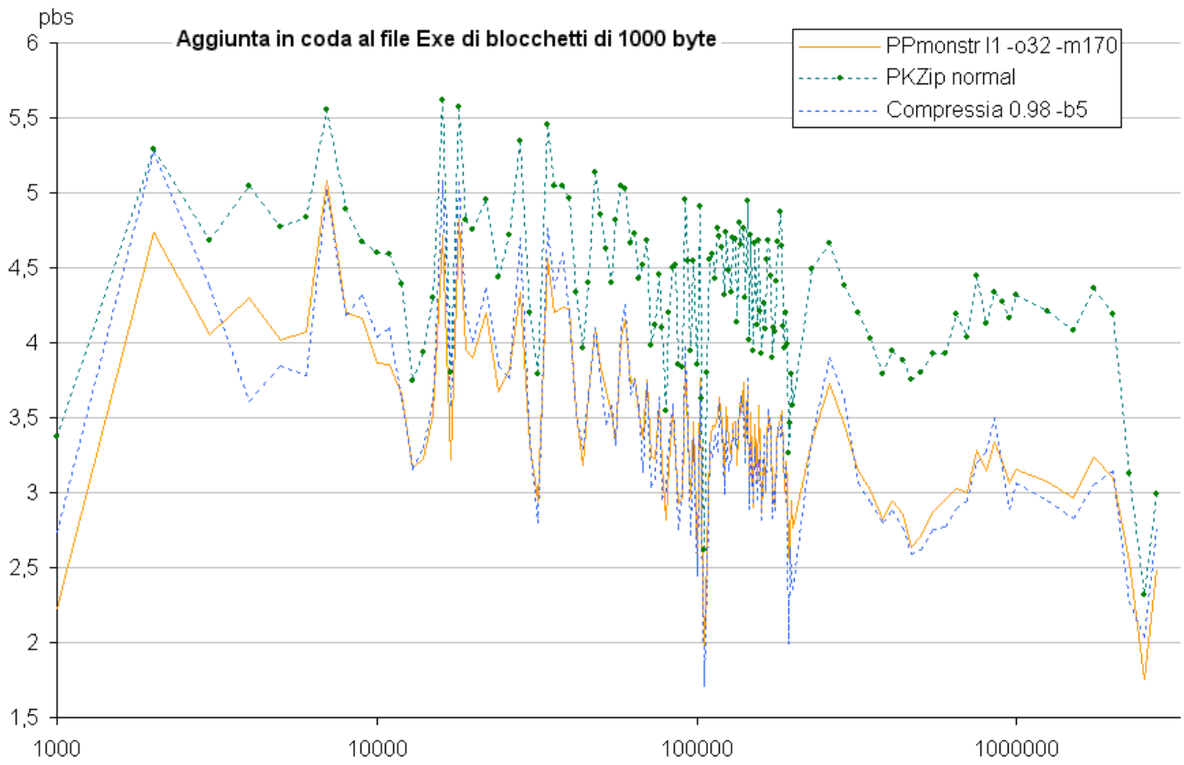
```
0..19999:       inc(LungOut, 1000);
20000..199999:  inc(LungOut, 2000);
200000..499999: inc(LungOut, 30000);
500000..999999: inc(LungOut, 50000);
else            inc(LungOut, 250000);
```

Successivamente per tutti i valori misurati ho calcolato:

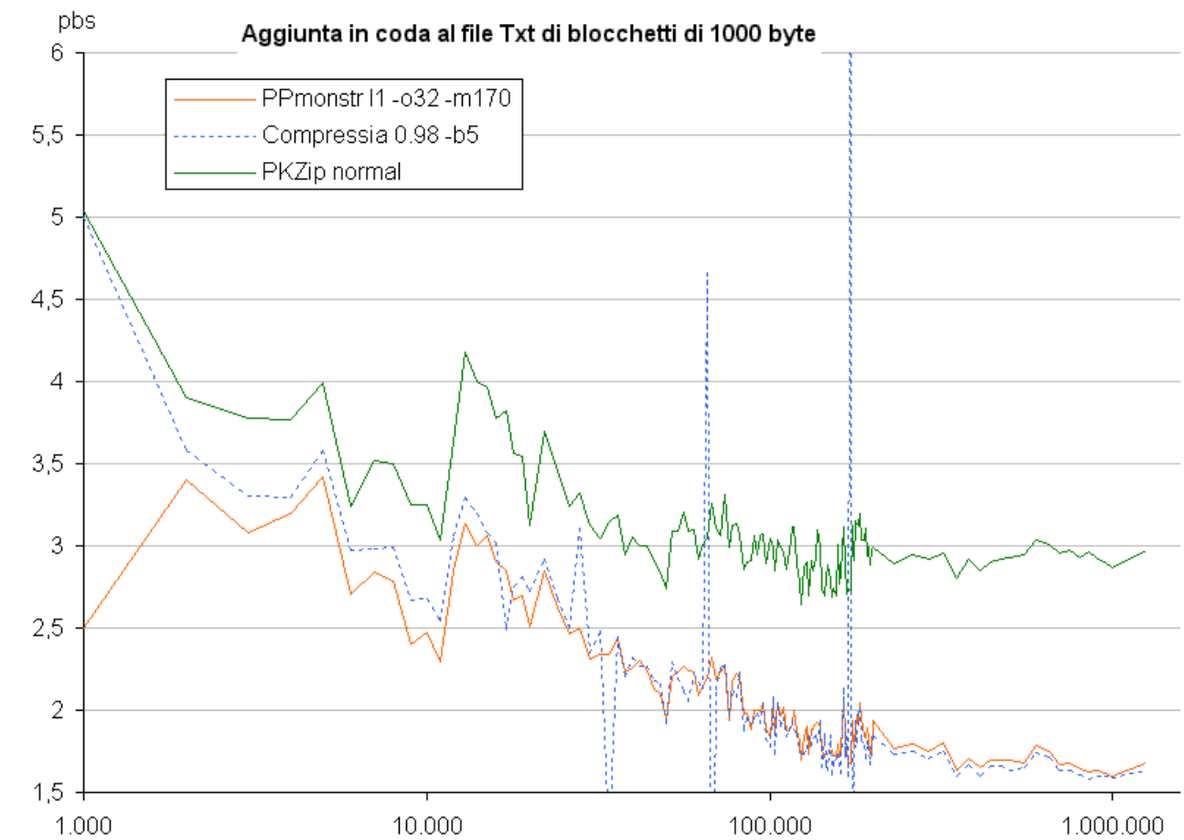
(LungCompressoAttuale - LungCompressoPrecedente) / (LungAttuale - LungPrec)

Dove:
  LungCompressoAttuale = Lunghezza del i-esimo file compresso
  LungCompressoPrecedente = Lunghezza del i-1-esimo file compresso
  LungAttuale = Lunghezza del i-esimo file da comprimere
  LungPrec = Lunghezza del i-1-esimo file da comprimere

Ecco i risultati di questa prova effettuata sul file Exe con tre compressori diversi, PPmonstr, Compressia 0.98 e PkZip. L'asse orizzontale è logaritmico in modo da permettere di osservare meglio e come calano i bps:



Come si sapeva già questo file non è uniforme, comunque il tasso bps sembra calare almeno fino a 200'000 byte.
La stessa prova l'ho effettuata anche col file Txt, con risultati interessanti:



Il PkZip (data la piccola base statistica su cui è fatto per lavorare) si stabilizza intorno ai 3 bps ben prima dei 100'000 byte. Compressia e Ppmonstr sono notevolmente appaiati, e sembrano mostrare che i bps calano fin oltre il milione di byte di base statistica. In realtà test fatti da altre persone potrebbero mostrare che il calo di bps prosegue ben oltre.
Si noti che il compressore basato su PPMII (PPmonstr) e quello basato sulla BWT (Compressia) funzionano su principi molto diversi. Il PPM è "incrementale", cioè comprime facenso una singola scansione del file dati in ingresso, a differenza del BWT. In un altro articolo presente sul mio sito ("Esperimento di compressione dati") ho mostrato come la base statistica del PPmonstr può essere

sfruttata per comprimere molto dei piccoli file statisticamente simili ad un grosso file che si dispone.
Col PPm è v
In prima approssimazione è vero che:
PPm(Base + Agg) = PPm(Base) + PPm(Agg | Stat(Base))

Dove:
PPm = compressione con PPmonstr
+ = incollaggio di un file in coda all'altro
Base = grosso file di cui si dispone statisticamente simile a Agg
Agg = piccolo file da comprimere
| = data la base statistica calcolata su

Gli algoritmi come la BWT invece ordinano tutti i dati in ingresso (spesso divisi a blocchi per motivi di tempo di calcolo e di occupazione di memoria) e quindi comprimono rimescolando tutti i dati. Questo fa sì che in generale comprimendo un grosso file ottenuto dalla unione di un grosso file Base ed uno piccolo Agg (statisticamente simili) non si ottiene un file compresso la cui gran parte inziale è identica al file ottenuto comprimendo solo Base (per maggiori spiegazioni si veda l'altro articolo che ho scritto, citato prima).
Un'altra cosa evidente nel grafico sono alcuni picchi negativi prodotti dal Compressia, che non mi so spiegare. Apparentemente in certo casi aggiungendo un piccolo file in coda, il file compresso risulta più piccolo! Il che mi pare impossibile, ma visto il modo molto particolare in cui funziona l'algoritmo BWT non posso escluderlo del tutto... Su questo servirebbero altri test.

Note:

[1] Si vedano gli articoli divulgativi di Brian Hayes, "On the Threshold", American Scientist, January-February 2003:
http://www.americanscientist.org/Issues/Comsci03/03-01Hayes.html
E sempre di Brian Hayes, "Can't Get No Satisfaction", American Scientist, March-April 1997:
http://www.americanscientist.org/issues/Comsci97/compsci9703.html
E anche un turorial molto interessante e non divulgativo, "Phase Transitions and Structure in Combinatorial Problems", di Carla P. Gomes, Tad Hogg, Toby Walsh e Weixiong Zhang (1.4MB e 700 KB):
http://www.cs.wustl.edu/~zhang/links/ijcai-search-tutorial-part1.pdf
http://www.cs.wustl.edu/~zhang/links/ijcai-search-tutorial-part2.pdf
Stringa di chiave per moriri di ricerca come Google: "phase transitions" "optimization problems".

[**Indice**]