Note sul Sudoku

Marco Liverani*

Dicembre 2005

1 Introduzione

Il gioco del Sudoku è un rompicapo giapponese che sta riscuotendo un successo notevole nel cosiddetto "grande pubblico": pur essendo un gioco di tipo combinatorio e dunque con un *background* logico-matematico, riesce ad appassionare anche quanti non hanno competenze specifiche in questo settore della matematica. Il motivo del successo credo sia tutto nella grande semplicità delle regole del gioco, affiancate dal fatto che effettivamente a fronte di regole assai semplici, la soluzione del problema è tutt'altro che banale e propone una sfida intellettuale divertente, in grado di dare qualche soddisfazione al giocatore vincitore. Sono convinto che alcuni dei problemi più belli della Matematica siano proprio quelli che tutti possono comprendere con poco sforzo e poche nozioni di base, ma che solo pochi (in alcuni casi ancora nessuno!) riescono a risolvere; spesso accade infatti che gli strumenti matematici necessari per risolvere problemi apparentemente assai semplici siano invece estremamente avanzati e sofisticati e dunque, proprio per questo, inaccessibili ai più.

In queste note cerco di riassumere i problemi algoritmici e combinatori interessanti che possono emergere da una analisi più attenta del gioco.

Il problema da risolvere in una partita di Sudoku può essere riassunto nei seguenti termini: riempire una griglia di 9×9 elementi, in modo tale che ogni riga, ogni colonna ed ognuna delle nove sotto-griglie 3×3 contenga le cifre da 1 a 9. In Figura 1 è riportato un esempio di griglia correttamente completata.

In termini appena più precisi possiamo dire che è data una matrice quadrata di 9 righe e 9 colonne in cui ogni elemento $m_{i,j}$ può assumere come valore un numero naturale compreso tra 1 e 9: $1 \le m_{i,j} \le 9$ per $i, j = 1, \ldots, 9$. La matrice inizialmente contiene solo alcuni elementi non nulli, mentre la maggior parte delle posizioni è vuota. L'obiettivo del gioco è proprio quello di completare la matrice, collocando gli elementi in modo tale da rispettare i seguenti vincoli:

1. per ogni $i = 1, \ldots, 9$ deve risultare $m_{i,h} \neq m_{i,k}$ per ogni $h, k = 1, \ldots, 9$, $h \neq k$ (tutti gli elementi su una stessa riga devono essere differenti);

*Sito web http://www.mat.uniroma3.it/users/liverani/, e-mail liverani@mat.uniroma3.it. Ultimo aggiornamento 4 dicembre 2005.

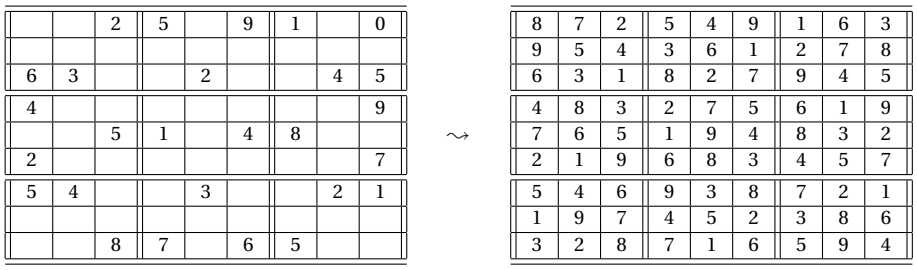

Figura 1: Una griglia di partenza e la soluzione finale

2. per ogni $j = 1,\ldots,9$ deve risultare $m_{h,j} \neq m_{k,j}$ per ogni $h,k = 1,\ldots,9$, $h \neq k$ (tutti gli elementi su una stessa colonna devono essere differenti);

3. per ogni $h = [i/3]+1,\ldots,[i/3]+3$ e per ogni $k = [j/3]+1,\ldots,[j/3]+3$ deve risultare $m_{i,j} \neq m_{h,k}$ (gli elementi di uno stesso riquadro 3×3 devono essere differenti; con $[x]$ indichiamo la parte intera di x).

Una configurazione di Sudoku è un caso particolare di *quadrato latino*: una matrice $n \times n$ i cui elementi sono tutti e soli gli elementi dell'insieme $\{1,2,\ldots,n\}$, tale che gli elementi di una stessa riga e di una stessa colonna siano tutti diversi. Due quadrati latini di ordine n si dicono *ortogonali* se le n^2 coppie formate dagli elementi corrispondenti dei due quadrati, sono tutte distinte. Un quadrato latino *normalizzato* è ottenuto ponendo la prima riga e la prima colonna uguali a $(1,2,\ldots,n)$: $m_{1,1} = 1, m_{1,2} = m_{2,1} = 2, m_{1,3} = m_{3,1} = 3, \ldots, m_{1,n} = m_{n,1} = n$.

Naturalmente, oltre a ragionare su eventuali strategie risolutive automatiche efficienti per vincere una partita di Sudoku, o su criteri tali da poter calcolare automaticamente la "mossa" migliore a fronte di una determinata configurazione di gioco, sono molti i problemi che possono essere derivati dal contesto suggerito dal gioco del Sudoku.

Ad esempio, un problema molto interessante ed assolutamente non banale è quello di stabilire il numero di possibili configurazioni differenti della scacchiera completata. Nell'affrontare la soluzione di questo problema, recentemente risolto da Felganhauer e Jarvis (vedi [1]), si deve innanzi tutto osservare che due matrici diverse, M ed M' possono invece rappresentare la stessa configurazione, nel caso in cui si possano rimappare tutti gli elementi dell'una sugli stessi elementi dell'altra; in altri termini le matrici M ed M' rappresentano la stessa configurazione se esiste un isomorfismo $\varphi : M \to M'$ tale che $\varphi(m_{i,j}) = \varphi(m_{h,k})$ se e solo se $m_{i,j} = m_{h,k}$. In Figura 2 sono riportate due configurazioni equivalenti: si può passare dalla prima matrice alla seconda con il seguente isomorfismo: $\varphi(8) = 1, \varphi(7) = 2, \varphi(2) = 3, \varphi(5) = 4, \varphi(4) = 5, \varphi(9) = 6, \varphi(1) = 7, \varphi(6) = 8, \varphi(3) = 9$.

Altre configurazioni equivalenti, possono essere ottenute per rotazione della matrice o per simmetria. Dunque le possibili configurazioni veramente differenti di Sudoku sono meno di quelle che potrebbero essere calcolate per semplice permu-

8	7	2	5	4	9	1	6	3
9	5	4	3	6	1	2	7	8
6	3	1	8	2	7	9	4	5
4	8	3	2	7	5	6	1	9
7	6	5	1	9	4	8	3	2
2	1	9	6	8	3	4	5	7
5	4	6	9	3	8	7	2	1
1	9	7	4	5	2	3	8	6
3	2	8	7	1	6	5	9	4

$\cong$

1	2	3	4	5	6	7	8	9
6	4	5	9	8	7	3	2	1
8	9	7	1	3	2	6	5	4
5	1	9	3	2	4	8	7	6
2	8	4	7	6	5	1	9	3
3	7	6	8	1	9	5	4	2
4	5	8	6	9	1	2	3	7
7	6	2	5	4	3	9	1	8
9	3	1	2	7	8	4	6	5

Figura 2: Due configurazioni equivalenti

tazione (pur nel rispetto di vincoli sopra elencati) degli elementi delle righe e delle colonne della matrice.

Calcolare il numero di quadrati latini di ordine n è molto difficile, tanto che tale risultato è noto solo per quadrati latini di ordine minore o uguale a 15. In generale esiste una relazione che lega il numero di quadrati latini di ordine n, $N(n,n)$, con il numero di quadrati latini normalizzati dello stesso ordine, $L(n,n)$:

$$N(n,n) = n!\,(n-1)!\,L(n,n)$$

Dunque il problema del calcolo del numero di quadrati latini differenti di ordine n, si riduce al calcolo del numero di quadrati latini normalizzati. Purtroppo non esiste una formula in grado di calcolare direttamente quel numero: al momento tale risultato, calcolato con algoritmi esaustivi (i cosiddetti metodi "a forza bruta", che si limitano a costruire e contare tutte le possibili configurazioni). Ad esempio $L(2,2) = 1$, $L(3,3) = 1$, $L(4,4) = 4$, $L(7,7) = 16.942.080$, $L(9,9) = 377.597.570.964.258.816$.

Partendo da considerazioni di questo tipo, ma restringendo il calcolo a quadrati latini con i vincoli ulteriori posti dalle regole del Sudoku, Felgenhuer e Jarvis in [1] hanno mostrato che il numero di configurazioni valide del Sudoku è pari a $2^7 \cdot 27.704.267.971 = 3.546.146.300.288$. È sorprendente notare che il fattore 27.704.267.971 è un numero primo.

Altri problemi interessanti, legati al gioco del Sudoku, possono essere riassunti nelle seguenti domande:

- Quale è il numero minimo di elementi che devono essere presenti nella configurazione iniziale per dare luogo ad una partita "non ambigua", ossia ad una partita con una sola soluzione?
- Quale è il criterio per cui, a parità di elementi iniziali non nulli, si può stabilire oggettivamente che una partita è più difficile di un'altra?
- Esiste un ordinamento (parziale o totale) delle partite di Sudoku in funzione della loro difficoltà?

2 Algoritmo ricorsivo per la ricerca esaustiva di una soluzione

Di seguito riporto un algoritmo ricorsivo per il calcolo della soluzione del Sudoku, a partire da una configurazione valida (ma incompleta) M. Indichiamo con $m_{i,j} = 0$ i valori non definiti della configurazione rappresentata dalla matrice M.

SudokuSolve(M)

```
1:  rc = 0
2:  siano i e j gli indici minimi per cui m_{i,j} = 0
3:  se esiste m_{i,j} = 0 allora
4:    per k = 1,2,...,9 ripeti
5:      sia m_{i,j} = k
6:      se M è una configurazione valida e SudokuSolve(M) = 1 allora
7:        rc = 1
8:      fine-condizione
9:    fine-ciclo
10:   se rc = 0 allora
11:     m_{i,j} = 0
12:   fine-condizione
13: altrimenti
14:   Sudoku risolto! Stampa la matrice M che rappresenta la soluzione
15:   rc = 1
16: fine-condizione
17: restituisci rc
```

Algoritmo 1: Algoritmo per il calcolo della soluzione a partire dalla configurazione M

La funzione ricorsiva SudokuSolve(M) restituisce 1 (*true*) se la configurazione M è compatibile, altrimenti restituisce 0 (*false*). Al termine della catena di chiamate ricorsive, quando tutti i valori della matrice sono stati assegnati in modo compatibile con le regole, la funzione stampa la matrice M che rappresenta la configurazione finale che risolve la partita.

Riferimenti bibliografici

[1] B. Felganhauer, F. Jarvis, *Sudoku enumeration*, pagina web all'indirizzo http://www.shef.ac.uk/~pm1afj/sudoku/, Giugno 2005.

[2] E. Russel, F. Jarvis, *Sudoku enumeration: the symmetry group*, pagina web all'indirizzo http://www.shef.ac.uk/~pm1afj/sudoku/sudgroup.html, Settembre 2005.

[3] E. W. Weisstein et al. *Sudoku*, From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Sudoku.html, 2005.

[4] E. W. Weisstein, *Latin Square*, From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/LatinSquare.html, 2005.