Problema del Room Assignment*

29 marzo 2005

1 Il problema

Dato n un numero pari di studenti, il programma si occuperà di assegnarli ad $n/2$ stanze in modo tale che i conflitti interpersonali siano ridotti al minimo.

Il fenomeno dei conflitti tra ogni coppia possibile di studenti è descritto mediante una tabella di “attrito”: il valore (i,j) della tabella indicherà tramite un numero quanto gli studenti i e j si detestano a vicenda. Va subito notato che questa tabella, che verrà implementata tramite una matrice, è simmetrica rispetto alla diagonale principale (le coppie $[i,j]$ e $[j,i]$ avranno lo stesso valore).

Risolveremo questo problema mediante il metodo di MONTECARLO e un algoritmo nota come *algoritmo di temperatura simulata*.

2 Metodo di MONTECARLO

Il metodo di MONTECARLO è descritto diffusamente (assieme al problema della generazione di numeri pseudocasuali in modo parallelo) in un'altra sezione del sito; si rinvia a quella per i dettagli.

3 Algoritmo di temperatura simulata

La temperatura fisica è il processo chimico di riscaldamento di un solido fino a quando esso non fonde: una volta fuso, esso viene poi raffreddato lentamente, in modo tale da ottenere un cristallo con una struttura regolare e livello energetico minimo. La *temperatura simulata* è un'applicazione del metodo di temperatura per la risoluzione ottimizzata di problemi di tipo combinatorio: una soluzione del problema è costituita dallo stato in cui si trova il materiale, il valore della funzione di valutazione per una particolare soluzione rappresenta l'energia associata a quel particolare stato, ed infine la soluzione ottimale corrisponde allo stato di energia minimo.

*Tratto da: Michael J. Quinn, *Parallel Programming in C with MPI and OpenMP*, McGraw Hill 2003

L'algoritmo di temperatura simulata è di tipo iterativo, ed inizia generando una soluzione iniziale (ottenuta casualmente o euristicamente) ed assegnando un valore al parametro t chiamato temperatura.

Durante ogni iterazione la soluzione corrente verrà casualmente cambiata per creare una nuova soluzione in prossimità della precedente: se la nuova configurazione ha una valutazione inferiore della precedente, essa viene allora accettata e diventa la soluzione corrente. In caso contrario la nuova soluzione verrà accettata con probabilità $e^{-\Delta/t}$, dove Δ è la differenza tra i valori delle due soluzioni e t è la "temperatura" corrente.

Perché mai dovremmo spostarci da una soluzione che è migliore di quella che abbiamo appena trovato? Il motivo sta nel fatto che le funzioni oggetto di studio tendono ad avere un elevato numero di massimi e minimi locali. Noi non vogliamo che il nostro algoritmo si fermi troppo presto e si blocchi su un minimo locale: nel momento in cui la temperatura è alta, dunque, sarà più facile per l'algoritmo uscire da un minimo locale, mentre quando la temperatura diminuirà la probabilità di fare altrettanto sarà nettamente inferiore per cui sarà più difficile uscire da un minimo profondo. In questa scelta entrano in gioco due fattori:

- la differenza Δ tra il valore della soluzione appena ottenuta ed il valore della soluzione corrente: ad una stessa temperatura, maggiore sarà questa differenza minore sarà la probabilità di accettare la nuova soluzione trovata;
- il valore corrente di t : più t sarà elevato, più sarà probabile che venga accettato un movimento in salita;

Per risolvere un problema di temperatura simulata dovremmo quindi:

- decidere come rappresentare le soluzioni
- definire una funzione di valutazione
- definire come generare una nuova soluzione nelle vicinanze di quella corrente
- realizzare una funzione di raffreddamento

Vediamo ora caso per caso come adattare questi quattro punti alla risoluzione del problema di assegnazione delle stanze descritto inizialmente.

3.1 Rappresentazione delle soluzioni

Una soluzione è l'assegnazione di n studenti in $n/2$ stanze.

Per tenere traccia di queste assegnazioni creeremo un array a , che conterrà gli interi compresi tra 0 ed $n/2-1$, e rappresenterà dunque per ogni persona la stanza a cui è stato assegnato. Va notato che ogni valore intero compreso tra 0 ed $n/2-1$ comparirà esattamente due volte nell'array a .

3.2 Funzione di valutazione

Partiamo dal considerare una matrice d , che conterrà dei numeri in virgola mobile compresi tra 0 e 10 che rappresenteranno il “*coefficiente d’attrito*” tra i vari studenti.

La funzione che valuterà la bontà della nuova configurazione dovrà preoccuparsi semplicemente di sommare le incompatibilità di ogni coppia di studenti che condividono la stessa stanza.

3.3 Generazione di una nuova soluzione

La generazione di una nuova soluzione è altrettanto semplice: basterà scegliere casualmente due studenti e scambiarli di stanza.

3.4 Funzione di temperatura

La scelta della funzione di temperatura avrà una incidenza profonda nelle performance del programma che ci accingiamo a sviluppare: più lentamente la temperatura si raffredda, più probabilità c’è di ottenere la soluzione ottimale, ma allo stesso modo il tempo di esecuzione si allungherà notevolmente e potrebbe divenire virtualmente infinito.

Una buona funzione di temperatura dovrà permettere di ottenere buone soluzioni a tempi computazionali accettabili.

Per questo problema utilizzeremo una funzione di temperatura geometrica:

$$t_0 = 1$$

$$t_{i+1} = 0.999t_i$$

4 Parallelizzazione

La temperatura simulata non garantisce di trovare una soluzione ottimale (letta dal punto di vista della funzione che viene esplorata, non garantisce di trovare il minimo assoluto della funzione).

Infatti, lo stesso algoritmo applicato utilizzando numeri casuali differenti può portare a soluzioni diverse. Questa riflessione ci porta a considerare utile una parallelizzazione del programma: lo stesso algoritmo verrà eseguito da diversi processi, ognuno dei quali effettuerà un *reseed*¹ del proprio generatore di numeri pseudo-casuali per ottenere una sequenza diversa da quella di ogni altro processo. Una volta conclusa l’esecuzione, verrà presa la migliore delle soluzioni dei vari processi.

Tutto questo garantisce una buona esplorazione della funzione (i cui termini però non possono essere determinati a priori) a fronte di tempi di esecuzione accettabili.

¹reseed: l’atto di cambiare i semi utilizzati da un generatore per generare numeri pseudo-casuali. Quest’azione comporta la restituzione di una sequenza di numeri differente dalla precedente.

5 Conclusione

Lo studio effettuato si conclude con l'implementazione dell'algoritmo in C tramite librerie MPICH2 (vedi link appropriato in questo sito).

Esso rappresenta sicuramente un buon esempio di fruizione dei vantaggi della programmazione parallela, e di applicazione dei metodi di Monte Carlo alla risoluzione di problemi combinatori.

29/03/2005, Enrico Sartorello (<enrico@sartorello.org>, <enrico@kpevillage.org>)