Resoconto dell'incontro interno obbligatorio

Scortegagna Carlo

6 febbraio 2009

1 Scopo dell'incontro

In data 6 febbraio 2009, alle ore 9.30, si è svolto un incontro tra i componenti del team QuiXoft con il preciso scopo di organizzare la parte algoritmica del progetto SIGEOL.

Nei giorni precedenti, in seguito a vari studi di fattibilità fatti dai singoli componenti del team di sviluppo, era stata scartata la proposta di sviluppare internamente l'algoritmo di calcolo dell'orario di lezione, data la sua complessità e dato l'elevato numero di algoritmi open source disponibili.

La presente riunione si prefigge quindi di scegliere l'algoritmo più adatto agli scopi del team QuiXoft e del progetto SIGEOL, valutando i pro e i contro delle varie proposte disponibili.

2 Membri presenti

Data l'importanza dell'argomento, sono presenti all'incontro tutti e 7 i membri del team QuiXoft.

3 Resoconto dettagliato

Durante l'incontro sono stati presi in considerazione svariati algoritmi open source ed ognuno ha presentato vantaggi e svantaggi. La scelta finale è stata presa valutando attentamente le seguenti caratteristiche:

- sono state valutate prima di tutto le effettive funzionalità dell'algoritmo preso in esame, ponendo l'attenzione che nessuna delle caratteristiche basilari richieste dal progetto SIGEOL fosse assente.
- è stata in seguito valutata la documentazione disponibile in allegato all'algoritmo: è infatti indispensabile avere a disposizione una documentazione chiara ed esaustiva per poter facilmente implementare l'algoritmo e usarlo per i nostri scopi.

- è stato posto un occhio di riguardo anche alle difficoltà che un dato algoritmo avrebbe potuto causare: sono stati, per esempio, preferiti algoritmi scritti in linguaggi di programmazione ben conosciuti ai membri del team QuiXoft e facilmente interfacciabili con il framework Ruby on Rails.
- infine, sono state anche valutate le performance degli algoritmi: è appunto preferibile avere un algoritmo che trovi delle buone soluzioni d'orario, possibilmente nel più breve tempo possibile. Le performance temporali, come da accordo con il comittente, non hanno comunque pesato più di tanto nella scelta finale dell'algoritmo.

A seguire la lista degli algoritmi valutati, comprensiva di una sintesi delle considerazioni fatte :

FET - *http://ilpassodellupo.it.gg/FET.htm*

E' scritto in C++, è quindi di facile comprensione per i membri del team rispetto ad altri linguaggi di programmazione. E' dotato di una buona documentazione e il codice è ampiamente documentato. L'algoritmo in se si basa su delle euristiche. Un problema è rappresentato dal fatto che l'algoritmo, essendo scritto in C++, dovrebbe essere preventivamente compilato sul server, limitando la portabilità del sistema.

Tablix - *www.tablix.org*

E' scritto completamente in C, il che rappresenta notevoli problemi di comprensione e di implementazione. C'è la possibilità di aggiungere vincoli non previsti aggiungendo moduli al kernel, che comunque risulta ben fornito. Si basa su un algoritmo genetico modificato. C'è anche la possibilità di distribuire il calcolo su più pc. Un problema è rappresentato dal fatto che l'algoritmo funziona solamente su piattaforme Unix.

UniTime - *www.unitime.org*

Algoritmo scritto in Java, quindi completamente ad oggetti. Utilizzando JRuby si possono chiamare metodi di Java dentro Ruby e viceversa. Questo rappresenta un notevole punto a favore per la portabilità del sistema. L'algoritmo si è dimostrato veloce nel trovare la soluzione. C'è la possibilità di impostare il tempo di esecuzione a piacimento. Punto da non trascurare, ha vinto il concorso ITC2007 (International Timetabling Competition, *www.cs.qub.ac.uk/itc2007*), dimostrandosi competitivo nei anche nei confronti di algoritmi e software commerciali.

Gstpl - *http://gstpl.sourceforge.net/*
Algoritmo genetico, il progetto sembra valido e ben documentato, ma non è più aggiornato da tempo. L'interfaccia grafica di prova in Java risulta molto instabile, non si sa se per colpa dell'interfaccia stessa o per colpa dell'algoritmo.

4 Decisione finale

Al termine della valutazione di tutti gli algoritmi sopracitati è stato scelto di utilizzare UniTime, per la comodità nell'avere un algoritmo scritto in Java e sfruttabile con Jruby e per le notevoli performance dimostrate.