B003725 Intelligenza Artificiale (2018/19)

Studente: Lorenzo Macchiarini (6129400) — *<2018-12-17 Mon>*

Elaborato assegnato per l'esame finale

Istruzioni generali

Il lavoro svolto sarà oggetto di discussione durante l'esame orale e dovrà essere sottomesso per email due giorni prima dell'esame, includendo:

1. Sorgenti e materiale sviluppato in autonomia (non includere eventuali datasets reperibili online, per i quali basta fornire un link);
2. Un file `README` che spieghi:
   - come usare il codice per riprodurre i risultati sottomessi
   - se vi sono parti del lavoro riprese da altre fonti (che dovranno essere **opportunamente citate**);
3. Una breve relazione (massimo 4 pagine in formato pdf) che descriva il lavoro ed i risultati sperimentali. Non è necessario ripetere in dettaglio i contenuti del libro di testo o di eventuali articoli, è invece necessario che vengano fornite informazioni sufficienti a *riprodurre* i risultati riportati.

La sottomissione va effettuata preferibilmente come link ad un repository **pubblico** su github, gitlab, o bitbucket. In alternativa è accettabile allegare all'email un singolo file zip; in questo caso è **importante evitatare di sottomettere files eseguibili** (inclusi files `.jar` o `.class` generati da Java), al fine di evitare il filtraggio automatico da parte del software antispam di ateneo!

Ricerca locale per problemi di sodisfacimento di vincoli

In questo esercizio si implementa una strategia di ricerca locale basata sull'euristica min-conflicts descritta in in R&N 2009 6.4. L'algoritmo può essere implementato in un linguaggio di programmazione a scelta. La base del codice dovrebbe essere sufficientemente generale da permettere di risolvere un problema generico ma le descrizioni del dominio e dei vincoli possono essere direttamente codificate nel programma (a differenza di quanto accadrebbe con un risolutore general purpose che sarebbe tipicamente in grado di leggere la specifica del CSP in un linguaggio formale). Il codice sviluppato dovrà essere testato su due problemi giocattolo: n-queens e map-coloring (per quest'ultimo è facile generare problemi casuali seguendo la strategia suggerita nell'esercizio 6.10 in R&N 2009). Si studi empiricamente il tempo di esecuzione del programma realizzato in funzione del numero di variabili n del problema, facendo crescere n quanto più possibile (nei limiti ragionevoli imposti dall'hardware disponibile).