Contents

1 Regine

1.1 Visto come problema di ricerca locale

1.2 Visto come problema di ricerca globale

questioni di ottimalità definite come il costo della sequenza di azioni ci interessano il giusto

il numero di scelte possibili sono $\binom{64}{8}$

altrimeni, sapendo che per le regole del gioco ci sarà una e una sola regina per riga, posso rappresentare lo stato come una tupla di 8 elementi, ognuno dei quali rappresenta una riga della scacchiera

in questo i possibili stati sono 8^8

quante sono le foglie di questo albero di ricerca?

1.2.1 Backtracking

il backtracking è una sorta di dfs che cerca di evitare lavori inutili, se un sottoalbero è maledetto, il sottoalbero non viene esplorato[1]

[1] si richiede comunque che il problema abbia una struttura tale da determinare "ma sto sottoalbero è tutto fottuto?"

è adatto per problemi cosiddetti di *soddifacimento di vincoli*, o CSP (Constraint Solving Problem), o CP[2] (Constraint Programming)

2 Declarative motherfucker

cos'è una soluzione, una soluzione può avere un costo

2.1 Definizioni

set of variables $X = \{x_1, x_2, x_3, x_4, x_5, \dots\ , x_n\}$

set of domains $D = \{d_1, d_2, d_3, d_4, d_5, \dots\ , d_n\}$

- qui d_i è il dominio su cui è definita la variablie x_i

set of constraints $C = \{c_1, c_2, c_3, c_4, c_5, \dots\ , c_n\}$

come definisco un constraint? Posso avere una funzione che $f : X \to \{0, 1\}$, questa sarebbe un po' un puttananio

uno strumento matematico utilizzabile potrebbe essere la *relazione* (la stessa di basi dati)

$$c_i := (scope,\ rel)$$

dove *scope* è un sottinsieme delle variabili, e rel è i valori validi per quella tupla

è simile all'idea di avere una funzione $f : X \to \{0, 1\}$, ma non è definita su tutto X, la definisci su una *proiezione* di X

- una **soluzione** è un assegnamento completo delle variabili $x_1, \dots, x_n$ che soddisfa tutti i vincoli

quindi se per ogni vincolo faccio la proiezione e poi ho la soluzione nella relaizione, allora ok, altrimenti sticazzi (le relazioni possono essere scritte come tabelle(metterle tutte), o come funzione(scrivo il check per cui questa sta nella relazione))

i problemi di soddifscimento di vincoli sono *NP-Completo*, vale a dire, so' cazzi (in genere, molte sottoistanze anche utilii sono parecchio trattabili)

le regine ammettono soluzioni polinomiali, ma il caso generico di problema di soddifscimento di vincoli, vale a dire il "ecco sti vincoli, risolvi il problema", non è risolvibile in tempo polinomiale

[2] non quello illegale

2.2 8 regine, alla declarative motherfucker

variabili gli indici delle regine nelle singole colonne

dominii $\{1, \ldots, n\} \forall$ *colonna*

2.2.1 E i vincoli?

- non ce ne sono due uguali, quindi $\forall\ i, j\ x_i \neq x_j$, noto anche come vincolo di $AllDifferent(x_1, \ldots, x_n)$, è uno dei vincoli standard
- non toccarsi in diagonale

il vincolo delle colonne è implicito nella modellazione del problema come vettore delle posizioni sulle colonne

2.2.2 Vincoli implicati

sono un po' un'alchimia aggiungere vincoli implicati o ridondanti alla specifica di un problema può far andare più veloce il solver

Si potrebbero aggiungere della variabili utilzzate per definire vincoli con queste, e boh.

2.2.3 Constraint propagation

con gli assegnameti i dominii si riducono, posso propagare e introdurre nuovi vincoli

la propagazione semplifica il problema, ma non lo risolve

2.3 Rete dei vincoli

grafo con

nodi variabili

archi se entrambe le variabili partecipano a un vincolo

3 Arc Consistency

X_i *AC wrt* X_j ovvero "X_i è Arc Consistent rispetto a (With Respect To) X_j" se $\forall$ valore $v_i \in D_i\ \exists$ valore $v_j \in D_j$ tale che la coppia (v_i, v_j) soddisfa il constraint **BINARIO** sull'arco (X_i, X_j), l'arco è orientato la relazione è asimmetrica, è un $\forall \in D_i \exists \in D_j$

3.1 Ipergrafi

sono tipo grafi, ma gli archi possono avere $1, 2, \ldots, n$ archi detta alla cazzo, hai dei nodi, e hai dei sottinsimei dei nodi, questi sottinsiemi sono gli archi, è un grafo generalizzato più di quanto il creatore avrebbe voluto

sono isomofri (vale a dire uguali) a database relazionali

3.2 Esempio (negativo)

- prendiamo 2 variabili, X e Y
- entrambi con dominio $\{0, \ldots, 9\}$
- prendiamo un solo constraint $Y = X^2$

è arc consistent? No ad esempio, per $X = 4$ non abbiamo valori di Y che soddisfano il constraint binario $Y = X^2$

se tutti gli archi nella rete sono consistenti, allora la rete è consistente

ok, mettiamo $D_x = \{0, 1, 2, 3\}$, la rete è consistente? Aspetta Y no! dobbiamo cambiare D_y per avere solo quadrati perfetti, ora la rete è consistente, le inconsistenze sono state abbattute, lunga vita alla consistenza, lunga vita al partito, amen

3.3 Algoritmo AC-3

AC sta per Arc Consistency[3], questo è il terzo

```
def ac_3(csp:Problem):
    arcs = Set(csp.get_all_arcs())
    while not arcs.is_empty():
        arc = q.extract()
        xi = arc.source()
        xj = arc.target()
        if revise(csp, xi, xj) == 'crossout done':
            if xi.get_domain().is_empty():
                return
            """
                            ok, fanculo tutto, abbiamo ridotto Di
                all'insieme vuoto,
                            questo problema non è risolvibile
                            ferma tutto, ferma tutto, MAYDAY! MAYDAY!!
                            """
            else:
                # ora che ho fatto il crossout devo rivedere potenziali
```

[3] di AC3 preferivo armoured core a dirla tutta

```
                # archi eliminati dal crossout che ho appena fatto
                for xk in csp.neighbours(xi):
                    if xk is xj:
                        # xj non ci interessa rimetterlo
                        continue
                    else:
                        arcs.insert(Arc(src=xi, dest=xk))
    return 'ok'

def revise(cps:Problem, xi:ProblemVar, xj:ProblemVar) -> bool:
    revised = False
    for x in xi.get_domain():
        if boh:
            """
                            if non c'è nessun y in xj.dominio() tale che
                tutti i
                            constraint binari tra x_j e x_j sono
                soddisfatti
                            """
            xi.get_domain().remove(x)
            revised = True

    return revised
```

l'assunzione di base è che i dominii siano finiti, se i dominii sono infiniti allora so' cazzi, l'assunzione è perchè un `for x in xi.get_domain():` non può operare su un dominio finito, questo algoritmo si addice maggiormente per solver di tipo `Finite Domain Solvers`