**Relazione su progetto per il corso di Ragionamento Automatico**

# Algoritmi di Path Consistency
# Confronto tra PC-1 e PC-2

**Autore : Cordioli Federico VR084495**

**Docente : dott. Farinelli Alessandro**

**Anno Accademico : 2009/2010**

# Basi teoriche

## Consistenza

Dati :

- Un insieme di **variabili**;
- Un **dominio** discreto per ogni variabile;
- Un insieme di **vincoli** su queste variabili.

Si avrà che ogni variabile assumerà tutti i valori appartenenti al proprio dominio discreto.
Per ogni coppia di variabili si avranno uno o più vincoli, ossia delle relazioni che possono limitare i valori assumibili dalla variabile all'interno del proprio dominio discreto.
Si avrà **consistenza** se tutti i vincoli sono rispettati dalle variabili e se per ogni variabile esisterà almeno un elemento assumibile all'interno del proprio dominio discreto.

## Grafo

Permette la rappresentazione di una rete di vincoli (costituita cioè da vincoli e variabili). In particolare dovrà essere non diretto e avrà come caratteristiche che :

- I nodi rappresentano le variabili;
- Gli archi rappresentano i vincoli tra le due variabili che collegano.

## Consistenza (consistency) di una rete di vincoli

Una rete di vincoli si dice **consistente** se esiste una assegnazione delle variabili che soddisfa tutti i vincoli della rete.

## Problemi di soddisfacimento vincoli

Se è necessario verificare la consistenza di un dato grafo si incorrerà in un problema di soddisfacimento di vincoli (in inglese abbreviato con **CSP**).

## Algoritmi per risoluzione CSP

Per risolvere problemi di soddisfacimento vincoli si utilizzano degli algoritmi che analizzando i vincoli presenti sugli archi del grafo e modificano la rete di vincoli per **forzare** la **consistenza.**
Nel dettaglio si tratta di estendere una consistenza locale. Ossia si selezionano insiemi di $i$-1 variabili a cui si faranno assumere tutti i valori che permettono di preservare la consistenza, quindi si estendono tali valori a $i$ variabili, e si procede cambiando gli elementi dell'insieme fino a raggiungere una consistenza globale dell'intera rete di vincoli o ha poter affermare che non è possibile ottenerla.
Le due principali tipologie di algoritmi per forzare la consistenza locale sono l'**Arc Consistency** e la **Path Consistency**.
In questa relazione si considererà la Path Consistency.
In questa tipologia di algoritmi la consistenza è ottenuta mediante la possibile eliminazione di tuple valide dalle relazioni, quindi limitando i rispettivi domini delle due variabili relazionate. Pertanto si potrebbe ottenere una rete di vincoli **diversa** dalla rete di partenza nel caso si siano **aggiunti o modificati dei vincoli.** Terminato l'algoritmo si potrà quindi asserire che qualunque valore assuma ogni variabile della nuova rete (all'interno del dominio fissato) si avrà comunque la consistenza.
Ne consegue che :
Consistenza => Path-consistenza
Path-consistenza $\not\Rightarrow$ Consistenza .

## Path Consistency

Mediante gli algoritmi di Path Consistency è possibile forzare la consistenza di un grafo.
Un grafo è Path-consistente se per ogni valore consistente di ogni coppia di variabili x,y può essere esteso ad ogni altra variabile z.
Ne consegue che un grafo non sarà Path-consistente se :

- per un qualsiasi x, y esiste una coppia di valori appartenenti al dominio non estendibile a z;
- per un qualsiasi x, y non esistono coppie di valori assumibili.

Sia quindi $R_{xy}$ l'insieme delle coppie di valori assumibili (relazioni) tra il nodo x e il nodo y, conoscendo i rispettivi domini e i vincoli che li legano.
È possibile forzare la Path Consistency mediante l'eliminazione di coppie di valori non estendibili dalla $R_{xy}$ , questo può però provocare la violazione della Path-consistenza sulle R adiacenti. Si rende quindi necessario riesaminare gli archi del grafo o perlomeno quelli strettamente correlati all'arco in oggetto.
Nel caso si arrivi a non avere più valori assumibile per una data coppia x,y si potrà dire con certezza che il grafo in oggetto non è Path-consistente.
Per forzare la Path-consistenza sul grafo si possono utilizzare vari algoritmi, quelle considerate in questa relazione sono PC-1 e PC-2, entrambe basate sull'utilizzo della revise3 che si occupa di individuare quali valori per la coppia in oggetto eliminare.

## Revise3

Cerca di estendere i valori assumibili dalle variabili analizzate x,y a una terza variabile z.
Per far ciò non toglie valori dal dominio bensì coppie dalle relazioni che legano x a y.
In particolare per ogni coppia ( $x_i$ , $y_i$ ) appartenente a $R_{xy}$ verifica se esiste un valore di z tale per cui ( $x_i$ , $z_n$ ) appartiene a $R_{xz}$ e ( $y_i$ , $z_n$ ) appartiene a $R_{yz}$ .
In pseudo-codice :

```
FOR ogni < a , b > appartenente a Rxy
        IF non esiste c app. a Dz t.c. ( a , c ) app. a Rxz e ( b , c ) app. a Ryz
                Cancella ( a , b ) da Rxy
        END IF
END FOR
```

## PC-1

La strategia di risoluzione **PC-1** può essere riassunta nel seguente pseudo-codice.

```
WHILE nessun vincolo è cambiato
        FOR k da 1 a n
                FOR tutti i , j  da 1 a n  tali per cui  i ≠ j ≠ k  e  i < j
                        Esegui revise3 (( xi , xj ), xk )
                END FOR
        END FOR
END FOR
```

## PC-2

La strategia di risoluzione **PC-2** può essere riassunta nel seguente pseudo-codice.

Coda Q ← {tutte le triple $< i , j , k >$ tali per cui $i < j$ e $k \neq i \neq j$ }
WHILE Q non è vuota
    Estrai $< i , j , k >$ da Q
    Esegui revise3 (( $x_i$ , $x_j$ ), $x_k$ )
    IF R su ( $x_i$ , $x_j$ ) sono cambiate
        Q ← Q U {$< l , i, j >$ , $< l , j , i >$ con $l \neq i \neq j$ }
    END IF
END WHILE

## Confronto tra PC-1 e PC-2

Già analizzando lo pseudo-codice è possibile trarre le prime considerazioni sulle differenze tra le due strategie.
Innanzitutto si nota che l'ordine iniziale con cui vengono analizzate le triple è il medesimo, vi sono però notevoli differenze nel caso in cui la revise3 rilevi la necessità di aggiornare l'insieme delle relazioni intercorrenti tra due variabili analizzate.
Se infatti PC-1, una volta conclusa l'analisi di tutte le triple, eseguirà nuovamente l'analisi di tutte le triple nel medesimo ordine; PC-2 invece aggiunge, se non già presenti, unicamente le triple correlate a quella appena analizzata.
Sebbene a prima vista PC-2 sembrerebbe quindi richiedere un numero di revise3 molto inferiore in ogni situazione (per il fatto di "aggiungere" un numero molto minore di triple da analizzare nel caso di un rilevamento) all'uso pratico il tempo di esecuzione potrebbe essere maggiore.
Questo perché il numero di triple da controllare in PC-1 è costante e indipendente da quanti aggiornamenti rilevi la revise3 all'interno del ciclo. In PC-2 il numero di triple da controllare varia invece dinamicamente ed è generalmente direttamente proporzionale al numero di aggiornamenti effettuati dalla revise3.
Inoltre mentre in PC-1 nei cicli successivi al primo l'ordine di analisi delle triple è lo stesso in PC-2 la coda Q conterrà le varie triple sempre in posizioni diverse ad ogni reinserimento e quindi si effettuano "salti" tra le varie regioni del grafo durante l'analisi.
La principale conseguenza di ciò è che mentre in PC-1 si ha una unica origine di "propagazione della consistenza" in PC-2 si hanno vari punti di propagazione.
Ne consegue che PC-1 potrebbe avere un tempo di computazione inferiore (sebbene risulti sempre richiedere un numero maggiore o uguale di revise rispetto a PC-2) nei casi in cui si hanno dei grafi caratterizzati da :

- elevato numero di aggiornamenti da parte di revise3 nel medesimo ciclo;
- un'unica "regione di non consistenza" iniziale.

PC-2 avrà in generale una maggior efficienza negli altri casi.

**Perché usare PC-1 allora?**
Perché fa uso di operatori più primitivi (vettore di interi e ciclo for) mentre PC-2 richiede necessariamente l'utilizzo di una coda il cui utilizzo potrebbe richiedere uno sforzo computazionale maggiore.
Quindi in generale è possibile affermare che se il numero di revise3 necessarie ai due metodi è circa lo stesso, PC-1 impiegherà un tempo di esecuzione minore.
Purtroppo essendo questa applicazione sviluppata per un sistema operativo general pourpouse il

tempo di esecuzione viene influenzato anche dall'esecuzione di altri task. Inoltre tale tempo dipende molto anche dal linguaggio in cui si è scritto e dal compilatore con cui si è compilato il codice. Ecco perché l'unico parametro di confronto oggettivo (e l'unico considerato nella nostra analisi) tra i due metodi è il numero di revise3 necessarie. Bisognerà comunque considerare che nel caso si ottenga che generalmente su un dato insieme di grafi simili le chiamate a revise3 tra le due strategie sono circa le stesse sarà preferibile usare PC-1.

# Implementazione
Si riportano ora le principali classi e tipi di oggetti dell'elaborato descrivendo funzioni e metodi implementati.

## Classi principali

## Main
Input : Ø.
Output : stampa a video il numero di revise3 richieste da PC-1 e PC-2.

Classe principale, si occupa di gestire il passaggio delle Relazioni tra il generatore e i metodi risolutivi. Inoltre stampa anche a video il confronto tra le esecuzioni dei due metodi.

## Generator
Input : genera, riceve da tastiera o legge da file il grafo da analizzare.
Output : ogniRxy (vettore contenente le relazioni del grafo).

Classe generatrice, si occupa di richiedere all'utente come desidera procedere al popolamento del grafo. É possibile :

- generare un insieme di relazioni sugli archi del grafo casuale. Specificando il numero di nodi, le dimensioni del dominio e il numero minimo e massimo di relazioni per gli archi ;
- digitare da tastiera il numero di nodi, le dimensioni del dominio e tutte le relazioni presenti sull'arco;
- specificare di leggere le caratteristiche del grafo direttamente dal file input.r0 .

Per ogni possibile scelta Generator si occuperà di creare un vettore di classe Relazione contenente quindi tutte le relazioni presenti sui vari archi.
Essendo un grafo fortemente connesso è possibile calcolare il numero di archi mediante la seguente formula : nodi*(nodi-1)/2 .
I vari archi saranno ordinati in ordine crescente in base al nodo sinistro (che per costruzione sarà sempre inferiore del nodo destro), e quindi in base al nodo destro che collegano.
Ad esempio nel caso si abbiano 4 nodi il vettore delle relazioni avrà gli archi nel seguente ordine : {(0,1); (0,2); (0,3); (1,2); (1,3); (2,3)} .
Questo permette di semplificare la trattazione dell'oggetto Relazioni oltre a diminuirne le dimensioni. Infatti non sarà necessario avere delle variabili interne all'oggetto Relazione per indicare :

- numero degli archi presenti sul grafo;
- numero dei nodi presenti sul grafo (ottenibile mediante la formula inversa);
- etichetta con il numero dell'arco;
- etichetta con i nodi corrispondenti all'arco.

Tutte queste informazioni sono ottenibili mediante la semplice analisi dell'oggetto Relazione pur non essendovi contenute.
Una volta ottenuto il vettore di classe Relazione verrà passato alla classe chiamante.

## PathC1
Input : ogniRxy.
Output : stampa a video il risultato dell'analisi e restituisce nRev (numero revise3 chiamate).

Classe che ricevuto in input il vettore di classe Relazione contenente tutte le relazioni presenti sul nodo esegue la PathC1 sul grafo.

Per estrarre i necessari valori dal vettore si servirà di un oggetto Etichetta che permette di individuare in che posizione nel vettore si trovano gli archi da passare alla Revise.
Effettua quindi le necessarie chiamate alla Revise3 monitorando l'ordine di analisi degli archi e se vengano o meno modificati dalla stessa.
Stampa quindi il risultato finale e ritorna alla classe chiamante il numero di Revise3 che sono state necessarie al suo ottenimento.

## PathC2

Input : ogniRxy.
Output : stampa a video il risultato dell'analisi e restituisce nRev.

Classe che ricevuto in input il vettore di classe Relazione contenente tutte le relazioni presenti sul nodo esegue la PathC2 sul grafo.
Per estrarre i necessari valori dal vettore si servirà di un oggetto Etichetta che permette di individuare in che posizione nel vettore si trovano gli archi da passare alla Revise.
Effettua quindi le necessarie chiamate alla Revise3 monitorando l'ordine di analisi degli archi e se vengano o meno modificati dalla stessa.
Stampa quindi il risultato finale e ritorna alla classe chiamante il numero di Revise3 che sono state necessarie al suo ottenimento.

## Revise

Input : R (contenente relazioni tra x e y) e Rz (contenente relazioni xz e yz).
Output : R (aggiornata).

Classe che ricevuto in input un oggetto Relazione da analizzare e un vettore contenente due oggetti Relazioni che le permettono di analizzarla esegue la Revise3 e ritorna il risultato.
Tutte le relazioni le sono passate già espresse nell'ordine corretto ossia rispettivamente :

- (x, y) ;
- (x, z) ;
- (y, z) .

Così che essa si debba occupare solo di controllare il contenuto e se necessario eliminare le coppie non estendibili.
La Revise ritornerà unicamente R (ossia le Relazioni tra x e y) sia nel caso non abbia eliminato niente sia nel caso abbia eseguito una o più eliminazioni. Si occuperà il chiamante a determinare indirettamente se la Revise ha apportato o meno un cambiamento e se questo cambiamento porterà ad avere un grafo Path-consistente o meno.

## Oggetti e classi accessorie

## Relazione

Input : x e y (contenenti il primo e il secondo elemento delle relazioni tra x e y).
Output : x o y.

É l'oggetto principale del progetto. Contiene al suo interno unicamente 2 vettori di interi, rispettivamente x e y.
Si è fatta questa scelta per limitare il più possibile le dimensioni dell'oggetto che dovrà essere passato alle varie chiamate di Revise3 così da diminuire il tempo necessario alla creazione e al popolamento dell'oggetto.

I due vettori conterranno i valori di tutte le coppie (x, y) ordinati secondo l'indice. Se si volesse quindi estrarre una coppia di valori bisognerebbe estrarre da entrambi i vettori l'elemento in posizione i-esima.
É stato scelto di utilizzare un oggetto formato da 2 vettori di interi anziché un unico vettore contenente oggetti formati da due int perché in Java la trattazione di vettori di tipo primitivo è maggiormente ottimizzata e quindi ci si aspetta di avere prestazioni migliori da questa impostazione. Inoltre questa scelta facilità la gestione delle inversioni necessarie a passare nell'ordine corretto le Relazioni alla Revise, sarà infatti sufficiente scambiare il puntatore dei due vettori senza doverli scorrere.
In esso oltre al costruttore che richiede unicamente il passaggio del vettore x e y, sono implementati anche i metodi per ritornare indipendentemente il vettore sinistro (x) e il vettore destro (y), oltre alla possibilità di invertire automaticamente i due vettori.

## Etichetta

Input : Archi (numero archi del grafo) o nodo1 e nodo2 (nodi di cui si vuole sapere posizione) o arco (posizione dell'arco).
Output : i (posizione arco) o eSx (nodo sinistro dell'arco) o eDx (nodo destro dell'arco).

É una classe che viene usata sia da Generator per il popolamento da file sia da PathC1 e PathC2 per estrarre le relazioni necessarie alla chiamata di Revise.
In pratica è formata da due vettori di dimensioni pari al numero di archi che contengono rispettivamente il nodo sinistro e il nodo destro di ciascun arco.
Vi sono quindi dei metodi per ottenere il numero di arco cercato dati i nodi e i nodi sinistro o destro conoscendo l'arco.

## Read

Input : digitazioni da tastiera.
Output : _int (intero digitato da tastiera).

Classe che permette di leggere da tastiera un intero.
Viene usata da Generator per ricevere la selezione dell'utente e per popolare il vettore nel caso lo stesso lo desideri.

## File di Input

File contenente il grafo espresso nella seguente forma.
AGENT a → indica il numero dei sotto-grafi presenti (nel nostro caso sarà sempre n=1).
VARIABLE n a d → specifica per ogni variabile :

- n → numero della variabile;
- a → sotto-grafo a cui appartiene;
- o → dimensioni del dominio.

CONSTRAINT l r → specifica che seguiranno i vincoli sull'arco tra :

- l → numero nodo sinistro;
- r → numero nodo destro.

NOGOOD x y → specifica le relazioni non accettate sull'arco specificato prima :

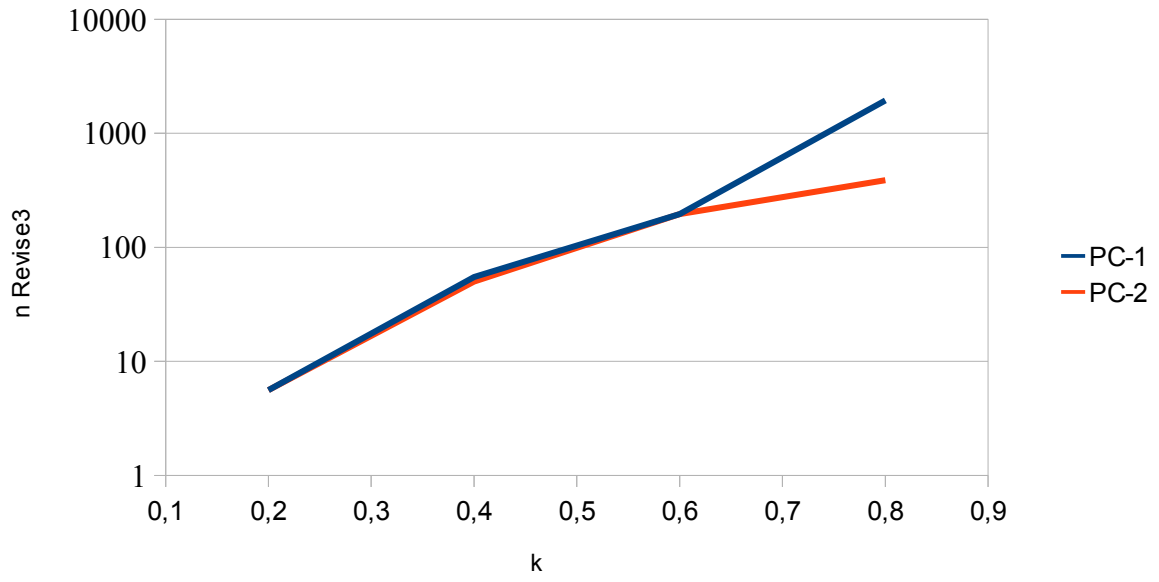
- x → valore non accettato su nodo l;
- y → valore non accettato su nodo r.

# Risultati esecuzione e Considerazioni

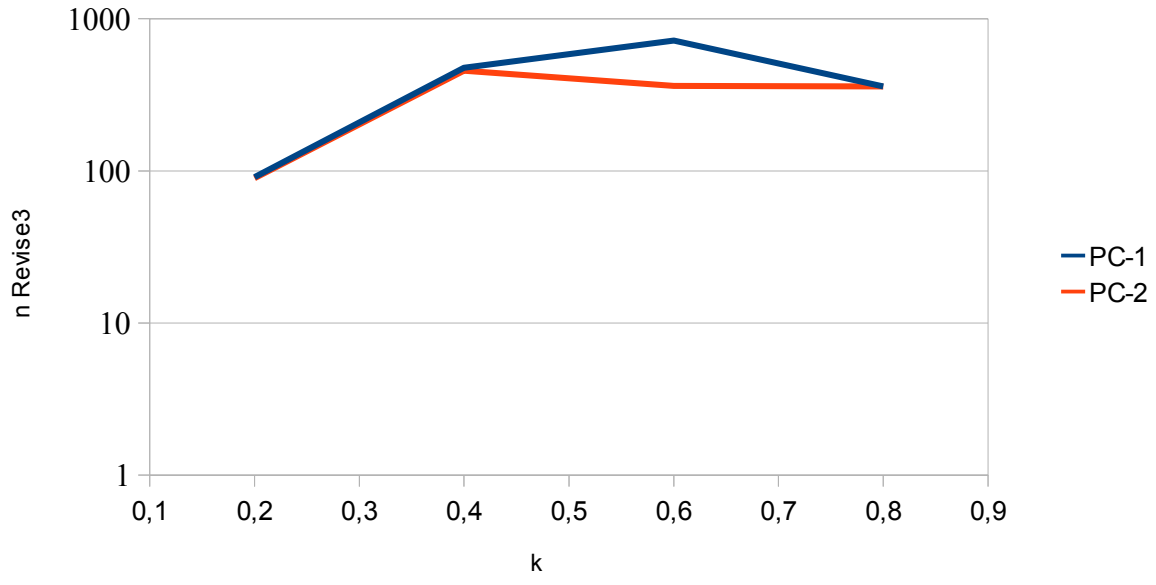
Si riportano ora i risultati ottenuti variando :

- il numero dei nodi del grafo;
- le dimensioni del dominio per ogni nodo;
- il coefficiente k che fissa il numero di relazioni per ogni arco, il numero di relazioni è fisso ed uguale per tutti gli archi e ottenuto dalla formula n = k * |dominio|^2 , con k compreso tra 0 e 1.

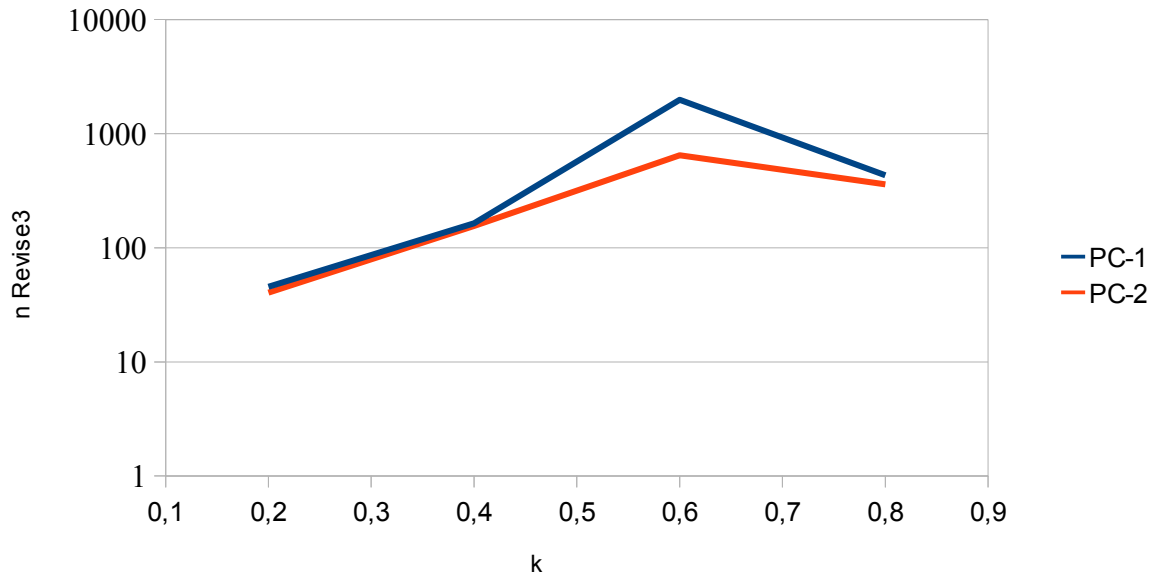
| Nodi → 10 | | | | Dominio → 5 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valore coefficiente e numero relazioni | | | | | | | | | | | |
| | k = 0,2 n = 5 | | | k = 0,4 n = 10 | | | k = 0,6 n = 15 | | | k = 0,8 n = 20 | | |
| | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. |
| Numero Revise3 | 5 | 5 | 0 | 47 | 47 | 0 | 171 | 171 | 0 | 2160 | 376 | 1784 |
| | 6 | 6 | 0 | 63 | 63 | 0 | 251 | 251 | 0 | 1800 | 404 | 1396 |
| | 4 | 4 | 0 | 58 | 52 | 6 | 212 | 212 | 0 | 2160 | 388 | 1772 |
| | 6 | 6 | 0 | 60 | 60 | 0 | 172 | 172 | 0 | 2160 | 403 | 1757 |
| | 7 | 7 | 0 | 46 | 29 | 17 | 173 | 173 | 0 | 1440 | 366 | 1074 |
| Media | 5,6 | 5,6 | | 54,8 | 50,2 | | 195,8 | 195,8 | | 1944 | 387,4 | |
| Max | 7 | 7 | | 63 | 63 | | 251 | 251 | | 2160 | 404 | |
| Min | 4 | 4 | | 46 | 29 | | 171 | 171 | | 1440 | 366 | |
| Dev. Std. | 1,02 | 1,02 | | 6,93 | 12,2 | | 31,7 | 31,7 | | 288 | 14,9 | |
| Diff. Max | | | 0 | | | 17 | | | 0 | | | 1784 |
| Diff. Min | | | 0 | | | 0 | | | 0 | | | 1074 |



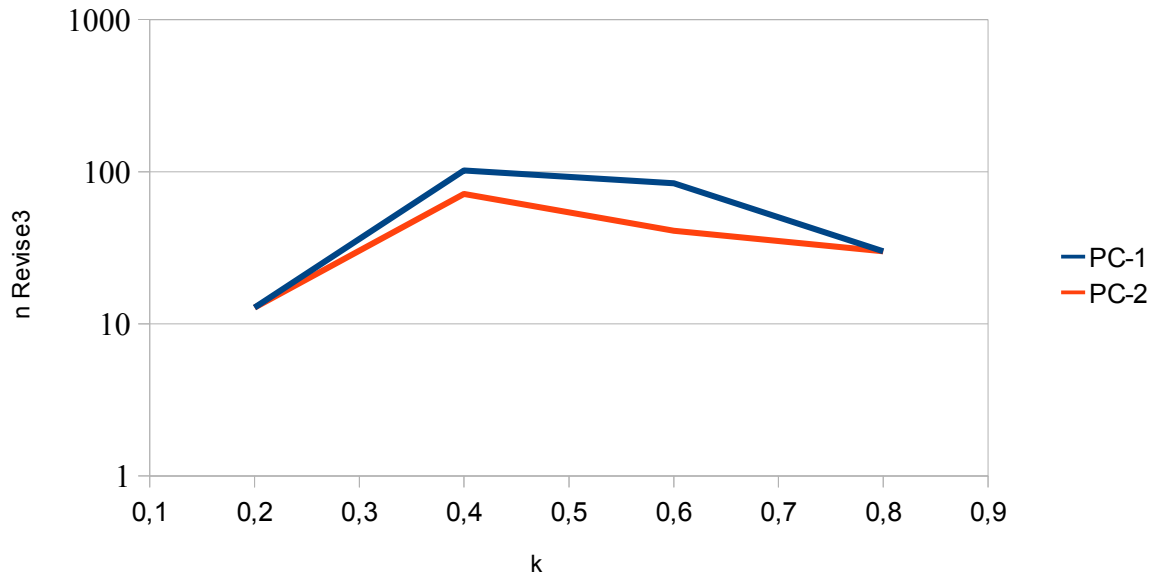
| Nodi → 10 | | | | | | | Dominio → 20 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valore coefficiente numero relazioni | | | | | | | | | | | |
| | 0,2 | | | 0,4 | | | 0,6 | | | 0,8 | | |
| | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. |
| | 92 | 88 | 4 | 450 | 432 | 18 | 720 | 360 | 360 | 360 | 360 | 0 |
| | 90 | 90 | 0 | 451 | 412 | 39 | 720 | 362 | 358 | 360 | 360 | 0 |
| | 91 | 90 | 1 | 495 | 491 | 4 | 720 | 364 | 356 | 360 | 360 | 0 |
| | 91 | 91 | 0 | 495 | 456 | 39 | 720 | 363 | 357 | 360 | 360 | 0 |
| | 92 | 89 | 3 | 490 | 489 | 1 | 720 | 366 | 354 | 360 | 360 | 0 |
| Media | 91,2 | 89,6 | | 476,2 | 456 | | 720 | 363 | | 360 | 360 | |
| Max | 92 | 91 | | 495 | 491 | | 720 | 366 | | 360 | 360 | |
| Min | 90 | 88 | | 450 | 412 | | 720 | 360 | | 360 | 360 | |
| Dev. Std. | 0,74 | 1,02 | | 21,1 | 31,1 | | 0 | 2 | | 0 | 0 | |
| Diff. Max | | | 4 | | | 39 | | | 360 | | | 0 |
| Diff. Min | | | 0 | | | 1 | | | 354 | | | 0 |



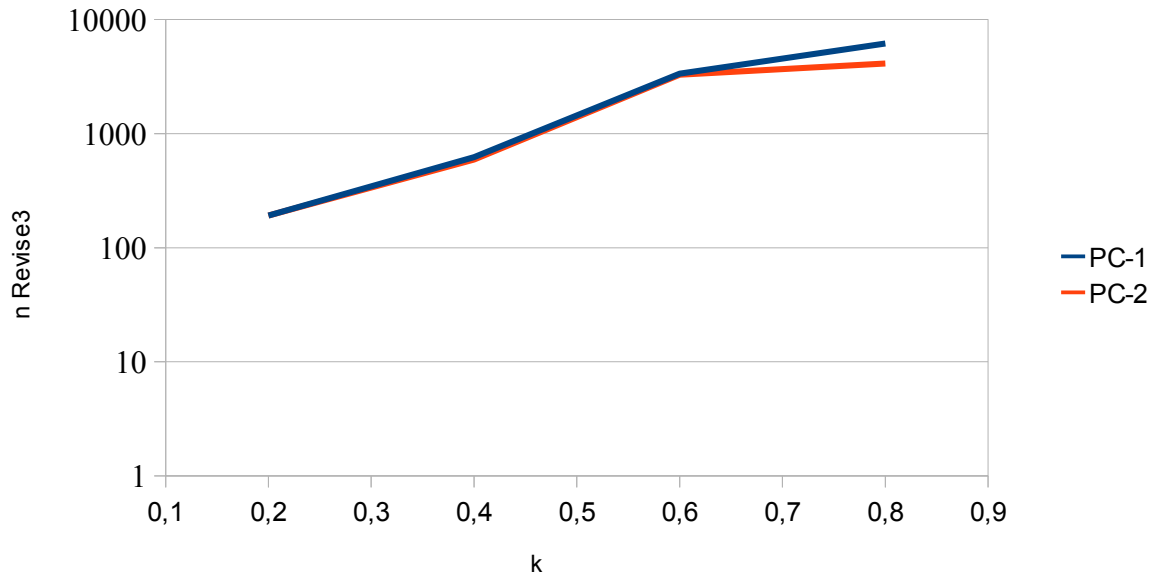
| Nodi → 10 | | | | | | | Dominio → 10 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valore coefficiente numero relazioni | | | | | | | | | | | |
| | 0,2 | | | 0,4 | | | 0,6 | | | 0,8 | | |
| | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. |
| Numero Revise3 | 46 | 46 | 0 | 132 | 132 | 0 | 1357 | 676 | 681 | 360 | 360 | 0 |
| | 46 | 46 | 0 | 171 | 171 | 0 | 1800 | 647 | 1153 | 360 | 360 | 0 |
| | 45 | 20 | 25 | 171 | 142 | 29 | 1346 | 659 | 687 | 720 | 360 | 360 |
| | 45 | 45 | 0 | 173 | 173 | 0 | 2520 | 637 | 1883 | 360 | 360 | 0 |
| | 45 | 45 | 0 | 171 | 159 | 12 | 2880 | 619 | 2261 | 360 | 360 | 0 |
| Media | 45,4 | 40,4 | | 163,6 | 155,4 | | 1980,6 | 647,6 | | 432 | 360 | |
| Max | 46 | 46 | | 173 | 173 | | 2880 | 676 | | 720 | 360 | |
| Min | 45 | 20 | | 132 | 132 | | 1346 | 619 | | 360 | 360 | |
| Dev. Std. | 0,49 | 10,2 | | 15,8 | 16,1 | | 620 | 19,3 | | 144 | 0 | |
| Diff. Max | | | 25 | | | 29 | | | 2261 | | | 360 |
| Diff. Min | | | 0 | | | 0 | | | 681 | | | 0 |



| Nodi → 5 | | | | | | | Dominio → 10 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valore coefficiente numero relazioni | | | | | | | | | | | |
| | 0,2 | | | 0,4 | | | 0,6 | | | 0,8 | | |
| | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. |
| Numero Revise3 | 10 | 10 | 0 | 90 | 75 | 15 | 90 | 41 | 49 | 30 | 30 | 0 |
| | 15 | 15 | 0 | 90 | 70 | 20 | 90 | 42 | 48 | 30 | 30 | 0 |
| | 18 | 18 | 0 | 120 | 70 | 50 | 90 | 39 | 51 | 30 | 30 | 0 |
| | 11 | 11 | 0 | 120 | 68 | 52 | 60 | 41 | 19 | 30 | 30 | 0 |
| | 10 | 10 | 0 | 90 | 75 | 15 | 90 | 42 | 48 | 30 | 30 | 0 |
| Media | 12,8 | 12,8 | | 102 | 71,6 | | 84 | 41 | | 30 | 30 | |
| Max | 18 | 18 | | 120 | 75 | | 90 | 42 | | 30 | 30 | |
| Min | 10 | 10 | | 90 | 68 | | 60 | 39 | | 30 | 30 | |
| Dev. Std. | 3,16 | 3,16 | | 14,7 | 2,83 | | 12 | 1,01 | | 0 | 0 | |
| Diff. Max | | | 0 | | | 52 | | | 51 | | | 0 |
| Diff. Min | | | 0 | | | 15 | | | 19 | | | 0 |



| Nodi → 20 | | | | | | | Dominio → 10 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Valore coefficiente numero relazioni | | | | | | | | | | | |
| | 0,2 | | | 0,4 | | | 0,6 | | | 0,8 | | |
| | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. | PC-1 | PC-2 | Diff. |
| Numero Revise3 | 190 | 190 | 0 | 623 | 598 | 25 | 3289 | 3289 | 0 | 6840 | 3420 | 3420 |
| | 196 | 196 | 0 | 587 | 587 | 0 | 3288 | 3288 | 0 | 6840 | 3421 | 3419 |
| | 192 | 192 | 0 | 573 | 566 | 7 | 3803 | 3483 | 320 | 3420 | 3420 | 0 |
| | 190 | 190 | 0 | 683 | 578 | 105 | 3250 | 3250 | 0 | 6840 | 6840 | 0 |
| | 190 | 190 | 0 | 642 | 642 | 0 | 3096 | 3081 | 15 | 6840 | 3420 | 3420 |
| Media | 191,6 | 191,6 | | 621,6 | 594,2 | | 3345,2 | 3278,2 | | 6156 | 4104,2 | |
| Max | 196 | 196 | | 683 | 642 | | 3803 | 3483 | | 6840 | 6840 | |
| Min | 190 | 190 | | 573 | 566 | | 3096 | 3081 | | 3420 | 3420 | |
| Dev. Std. | 2,24 | 2,24 | | 39,4 | 26,1 | | 240 | 128 | | 1368 | 1367 | |
| Diff. Max | | | 0 | | | 105 | | | 320 | | | 3420 |
| Diff. Min | | | 0 | | | 0 | | | 0 | | | 0 |



Nelle tabelle riportanti i risultati si sono evidenziate le esecuzioni che hanno prodotto un risultato Path-consistente ( ) .
Nei grafi si sono riportate il numero di esecuzioni necessarie al variare del coefficiente k per PC-1 e PC-2, evidenziando anche il valore minimo e il valore massimo ottenuto per ogni valore analizzato.
Dai dati sperimentali ottenuti possiamo affermare che :

- il numero di revise3 utilizzate da PC-2 è sempre minore o uguale al numero di PC-1;
- se la non path-consistenza (o la path-consistenza) viene rilevata durante il primo ciclo PC-2 e PC-1 necessitano del medesimo numero di revise3;
- se la non path-consistenza viene rilevata dopo il primo controllo degli archi PC-2 necessita un numero di revise3 minori o uguali a PC-1;
- superato un determinato valore di k tutti i grafi generati casualmente risultano consistenti;
- il valore del k necessario a ottenere grafi tutti consistenti aumenta all'aumentare del rapporto tra nodi e dominio;
- l'andamento della curva risultante evidenzia una fase crescente (in cui generalmente i grafi non sono path-consistenti) seguita da una decrescente (in cui invece sono generalmente path-consistenti). Quindi il numero di Revise3 all'aumentare di k cresce fino ad arrivare a un valore per cui è necessario effettuare un notevole numero di Revise3, superato tale picco tale numero decresce;
- le differenze maggiori tra le esecuzioni di PC-1 e PC-2 si hanno in prossimità di tale picco;
- PC-1 ha in generale un variabilità maggiore nel numero di esecuzioni necessarie rispetto a PC-2;
- mantenendo fisse le dimensioni del dominio si nota una crescita del valore di k, corrispondente al picco, all'aumentare del numero di nodi;
- mantenendo fisso il rapporto tra il numero di nodi e le dimensioni del dominio si nota una forte similitudine tra le curve risultanti, sebbene il picco si trovi per un valore di k minore nel caso si abbia un numero di nodi minore.

Ovviamente queste sono osservazioni effettuate su un numero di prove abbastanza contenuto, sarebbe necessaria una analisi più approfondita per poterle confermare con certezza.

Volendo sintetizzare è possibile ipotizzare che la scelta più conveniente (ricordando che PC-1 ha tempi di esecuzione inferiori richiedendo un numero di Revise3 circa pari a quello di PC-2) sarà :

- PC-1 nei casi in cui si debbano testare grafi principalmente non consistenti se il coefficiente k è basso;
- PC-1 nei casi in cui vi siano da testare grafi principalmente consistenti se il coefficiente k è alto ;
- PC-2 in tutti gli altri casi.