Pedigree

Documentazione aggiuntiva
Corso di reperimento dell'informazione
a.A. 2005/2006
prof.sa Maristella Agosti

Argenton Matteo
Buzzi Lorenzo
Gatto Giorgio
Molinaro Matteo
Zorzan Emmanuele

1 Prestazioni

Beagle è un'applicazione che appartiene alla categoria delle utilità di *Desktop Search*; è quindi progettato per il reperimento di documenti eterogenei, dai file di testo a quelli multimediali, dalle immagini ai file audio su un file system locale. Dovendo operare su una collezione limitata e definita dall'utente, il sistema di query di Beagle è stato costruito per operare con stringhe composte da pochi termini; lo strumento, infatti, utilizza un modello di reperimento booleano ed esegue un AND-filling implicito tra le parole dell'interrogazione. Di conseguenza stringhe troppo lunghe, avendo un'alta specificità, risultano troppo restrittive portando spesso a un reperimento vuoto.

Alla luce di queste osservazioni, acquistano un altro significato i risultati ottenuti utilizzando le query in linguaggio naturale fornite: i valori numerici sembrerebbero suggerire una scarsa capacità di reperimento da parte di Beagle. In realtà, poichè raramente un documento della collezione di test contiene tutti i descrittori della query, il numero di elementi restituiti è estremamente basso, quando non nullo. Questo comporta che un'alta percentuale di misure non sia calcolabile avendo denominatore pari a zero.

A supporto di quanto detto si è scelto di rieseguire il test utilizzando un OR-filling esplicito per ogni query. Come ci si aspettava il richiamo cresce notevolmente a discapito della precisione ma le medie, questa volta, sono calcolate su quasi la totalità delle interrogazioni.

Parametro	AND	#queries	OR	#queries
P_{AVG}	0.16	9	0.01	64
R_{AVG}	0.02	52	0.11	52
F_{AVG}	< 0.01	64	0.03	64

Questi primi risultati confermano le ipotesi, perciò si è deciso di approfondire l'indagine per aumentare la qualità e la confrontabilità (con gli altri gruppi) dei dati ottenuti. Si è proceduto, quindi, al calcolo di:

- Precisione con e senza le query nulle;
- Richiamo con e senza le query nulle;
- Fall-out con e senza le query nulle;
- Precisione Macro e micro con Cut-Off 10;
- Richiamo Macro e micro con Cut-Off 10;
- Fall-out con Cut-Off 10;
- Richiamo su Precisione;

Ogni voce è stata calcolata sia sui dati ottenuti con AND-filling che su quelli con OR-filling. I risultati sono stati riassunti nei grafici disponibili nella sezione allegata "Grafici dei Risultati".

Osservando i grafici emerge la seguente riflessione: come ci si aspetta, precisione e richiamo hanno andamento inverso in entrambi i casi (AND o OR) ma i valori di *precisione* e *richiamo* nel caso AND hanno, nella parte iniziale dei grafici, media molto elevata (a

differenza del caso OR).
Tale comportamento si può spiegare considerando che il primo documento reperito con AND-filling, se esiste, ha alta probabilità di essere rilevante in quanto contiene tutti i descrittori della query (alta specificità); questo vale anche per i successivi.
Viceversa una query completata con OR, ritorna un insieme più ampio di documenti dove, però, non è garantita la rilevanza del primo/i. Ciò si riflette sul valore medio che risulta conseguentemente più basso.

2 Configurazione

Beagle è un tool di IR scritto in C# e basato su DotLucene, un porting C# della libreria Lucene. Necessita perciò, essendo stato sviluppato per ambienti Linux, della virtual machine `mono` che è un porting del .NET Framework di Microsoft. Inoltre, pur potendo essere eseguito su qualsiasi Desktop Environment o Window Manager, Beagle utilizza le librerie dell'ambiente grafico Gnome, che deve essere installato in ogni caso.

Alla base del funzionamento di Beagle si trova il *daemon* `beagled`, un programma eseguito in background che si occupa sia dell'indicizzazione che del reperimento.
Il programma `beagle-search`, invece, è un'interfaccia grafica interposta tra l'utente e `beagled`, e che per esigenze di progetto non è stata utilizzata: se ne è realizzata una ad hoc di tipo web.
Il *daemon* crea, nella home dell'utente che lo lancia, la directory `.beagle`. Questa contiene i file di configurazione dell'utente, quelli contenenti gli indici e il log dell'applicazione.

Durante l'esecuzione, Beagle tiene traccia della propria attività in tre appositi file di log contenuti in `/home/username/.beagle/Log`:

- `current-Beagle` che mantiene traccia delle informazioni di avvio, delle operazioni compiute e di tutte le query esaminate;
- `BeagleExceptions` in cui vengono salvati gli output di errore;
- `BeagleIndexHelper` in cui viene tenuta traccia di tutti i file indicizzati.

Naturalmente ogni utente avrà i propri file di log (essendo memorizzati nella home directory).

Per ogni *backend*[1] Beagle memorizza un indice separato, perciò la risposta all'interrogazione viene data gestendo una sorta di *multicollezione* nel senso che è possibile scegliere che tipo di documenti reperire (tutti, solo e-mail, solo file di testo, solo immagini ecc...). Una volta scelto il tipo di documento non sono possibili ulteriori distinzioni in sotto-collezioni poichè, all'interno di ogni singolo backend, l'indice è unico.

I file di configurazione di ogni utente, infine, sono memorizzati nella sotto-directory `.beagle/config` in quattro file XML con codifica UTF-8.

3 Caratteristiche

Il sistema presuppone che la collezione, almeno per quanto riguarda i documenti di testo, sia monolingua poichè sia la stop list che le procedure di stemming sono efficaci solo per

[1] Beagle può indicizzare e-mail scaricate con diversi client di posta elettronica oltre ai documenti in molteplici formati presenti nel filesystem. Per ogni client, il daemon dispone di una componente specifica per l'indicizzazione e la ricerca. Tali componenti prendono il nome di *backend*.

l'inglese.
Per quanto riguarda i file multimediali (immagini e audio) il dizionario viene creato indicizzando i nomi di tali file.

Per quanto riguarda i propri file (log, indici, ecc...) Beagle utilizza la codifica UTF-8; questa viene assunta anche per i file di puro testo (txt, sorgenti, ecc...). Questo accade perchè `mono`, se non diversamente specificato, assume UTF-8 come codifica predefinita per l'I/O su file.

Gli sviluppatori non dichiarano esplicitamente limiti relativi alla taglia o al numero massimo di documenti indicizzabili. Beagle, inoltre, non riconosce archivi compressi (gzip, bzip2, ecc...).

Ogni utente del sistema che voglia utilizzare Beagle deve configurarlo specificando quale parte del filesystem indicizzare (avendone i permessi) e quali backend utilizzare.

L'interfaccia predefinita `beagle-search` presenta il risultato della query come serie di link agli elementi reperiti, sotto forma di icone. In particolare, selezionando l'icona corrispondente ad un risultato di tipo testuale viene aperto un form a piè pagina che visualizza la parte del documento contenente i termini cercati, evidenziati in grassetto.

Come già accennato nel paragrafo 1, la "query tipica" ipotizzata per Beagle è composta da due o tre parole. Per questo motivo il dizionario è semplicemente basato su *keyword*; questo gli conferisce dimensioni contenute che permettono il suo caricamento in memoria centrale. In ogni caso una copia del dizionario è salvata in memoria di massa nella directory `/home/username/.beagle/Indexes`. Nel caso in esame il dizionario occupa circa 2MB. A valle del dizionario si trova il posting file, realizzato con una struttura dati di tipo *B-tree*.

È possibile scegliere tra tre diversi metodi di ordinamento dei risultati ottenuti:

- `data di ultima modifica`
- `alfabetico`
- `rilevanza`

I primi due non hanno bisogno di ulteriori spiegazioni, mentre per il terzo è necessario specificare che viene utilizzato il sistema `term frequency`. Questo si basa sul numero di occorrenze dei descrittori estratti dalla query e presenti nel documento: più tale numero è elevato più il documento sarà rilevante.

Il sistema non realizza meccanismi di retroazione, nè espliciti nè impliciti. Nonostante sia possibile specificare query di tipo avanzato utilizzando gli operatori booleani non sono previsti meccanismi di query expansion veri e propri.

Beagle è dotato sia di un sistema di stemming che di una stop list (limitati alla sola lingua inglese) i quali permetterebbero di specificare query in linguaggio naturale. Purtroppo il modello di reperimento utilizzato, unitamente al sistema di pesatura, precludono tale possibilità a causa della presenza di troppi descrittori nella query (cfr. paragrafo 1).

Il servizio di `Help in linea` è fornito esclusivamente via web mediante un link all'indirizzo http://www.beaglewiki.org, dove comunque il supporto è limitato a chiarimenti su poche caratteristiche di base del programma.

Sono previste alcune forme di personalizzazione di `beagle-search` limitatamente alla collezione; è infatti possibile scegliere se indicizzare l'intera `home directory`, parte di essa o altri punti del filesystem, escludendo eventualmente alcune sub-directory.

4 Interfaccia Web

Il sistema di test realizzato si basa su tre macro-blocchi:

- `mono`: l'ambiente `mono` è la `virtual machine` che esegue tutto il codice C#, ovvero l'intero package di Beagle (`beagled`, `beagle-search`, ecc...) e l'interfaccia a linea di comando realizzata per interrogare il `daemon` dall'interfaccia web.
- `Apache2 + PHP`: l'interfaccia web è realizzata interamente in PHP e presenta una semplice pagina iniziale che mette a disposizione tre opzioni.
  - effettuare una query qualsiasi;
  - effettuare una query tra quelle di test;
  - effettuare l'intero test con le 64 query proposte e ottenerne i risultati.

  Per le ultime due è possibile scegliere se utilizzare l'AND-filling predefinito oppure l'OR-filling esplicito.
  L'interprete PHP è caricato come modulo dal web server Apache2.
- `MySQL`: questo sotto-sistema mantiene un semplice database di tre tabelle.
  - `Query`: contiene i testi delle 64 query di prova.
  - `Qrels`: contiene, per ogni query, l'elenco degli identificatori dei documenti rilevanti.
  - `DocLookup`: mantiene la corrispondenza tra gli identificatori dei documenti e i nomi dei file corrispondenti.

  L'accesso a tale database è consentito esclusivamente al codice dell'interfaccia web.

Per il test dell'applicazione è stata scelta la distribuzione Linux Ubuntu 5.10. Tale scelta è stata dettata da diversi fattori:

- semplicità di installazione;
- interfaccia user-friendly;
- semplicità di aggiornamento (il gestore dei pacchetti `apt` si occupa del download e della risoluzione automatica delle dipendenze);
- supporto nativo di Gnome (che è il Desktop Environment predefinito).

4.1 Funzionamento

Il codice php della pagina web comunica con il daemon `beagled` mediante un programma di interfaccia. Questo è scritto in C# e viene eseguito su `mono` in una shell `bash`; si occupa di passare la stringa di query a Beagle e restituire al php i risultati ottenuti.

Per gli obiettivi di benchmark del progetto, l'interfaccia è stata dotata della possibilità di effettuare un confronto dei risultati ottenuti con quelli attesi in modo da poter valutare quantitativamente le prestazioni, in termini di efficacia.
Quando si esegue una query predefinita o l'intero benchmark, per ogni risultato ottenuto php interroga il database `MySQL` (che contiene i risultati attesi) a scopo di confronto.

Grafici dei Risultati

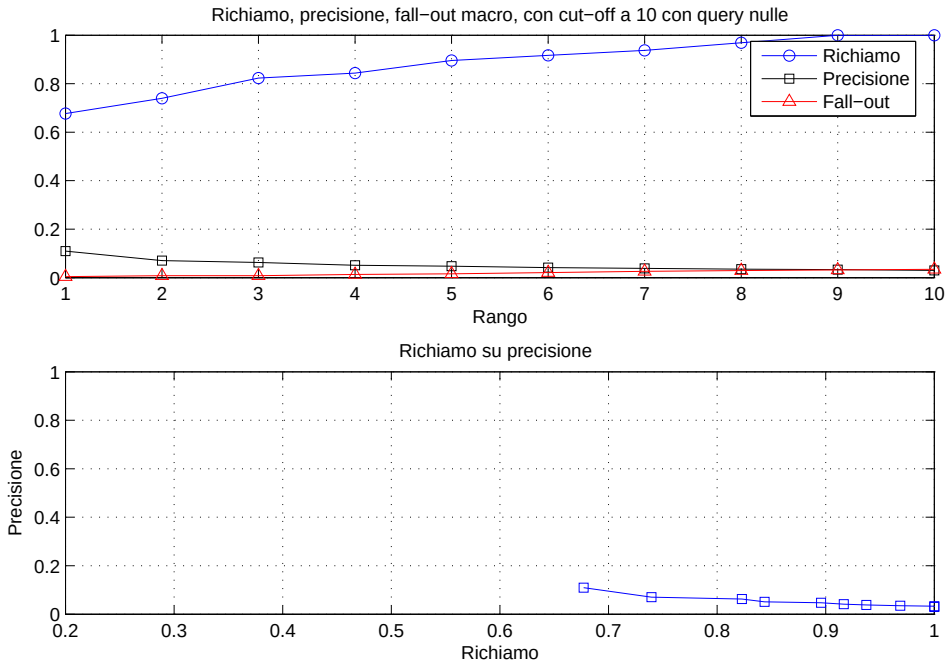

Figure 1: Benchmark con AND-filling

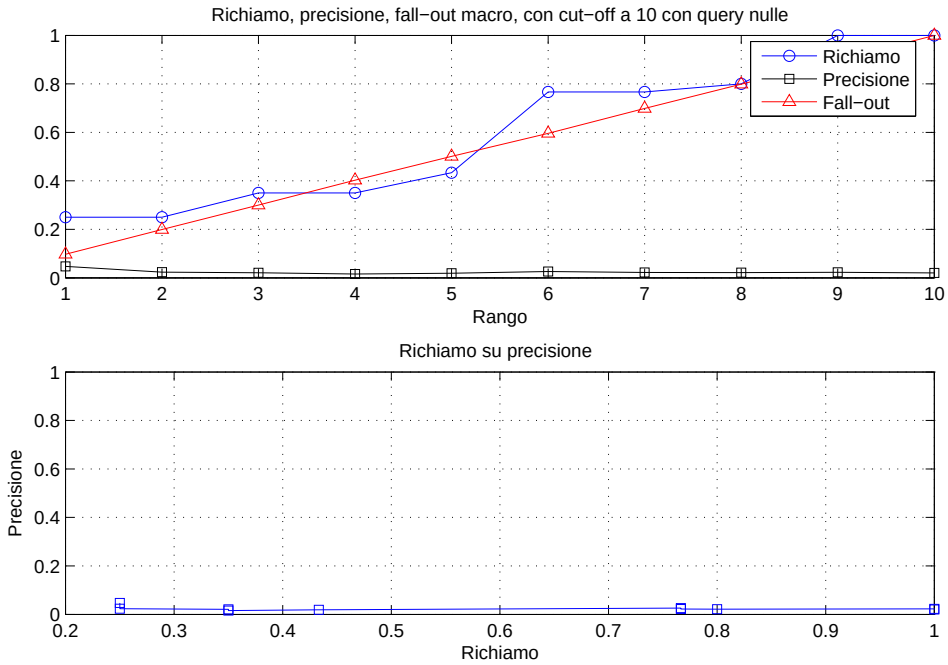

Figure 2: Benchmark con OR-filling

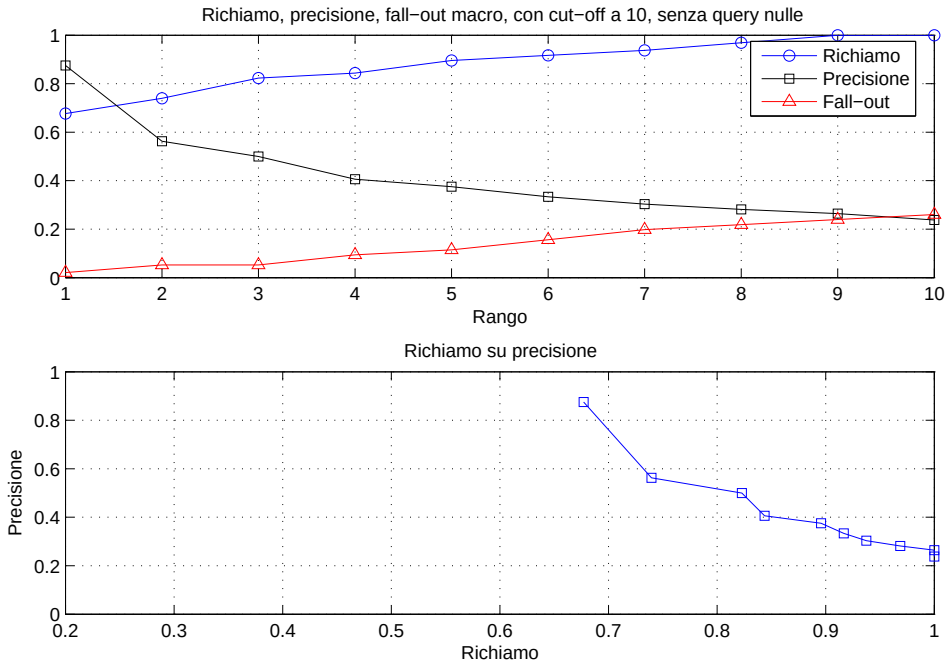

Figure 3: Benchmark con AND-filling

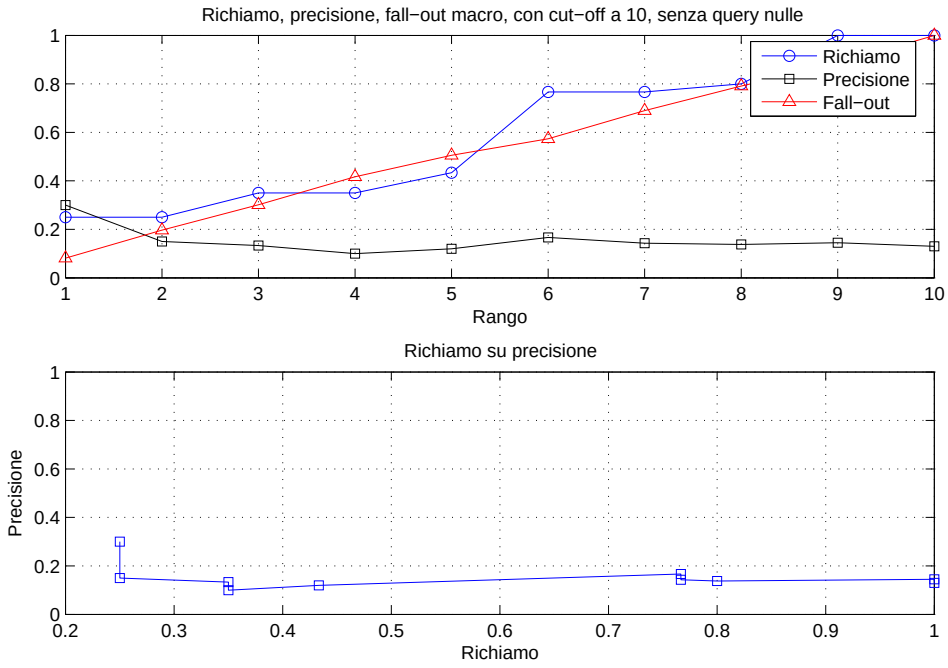

Figure 4: Benchmark con OR-filling

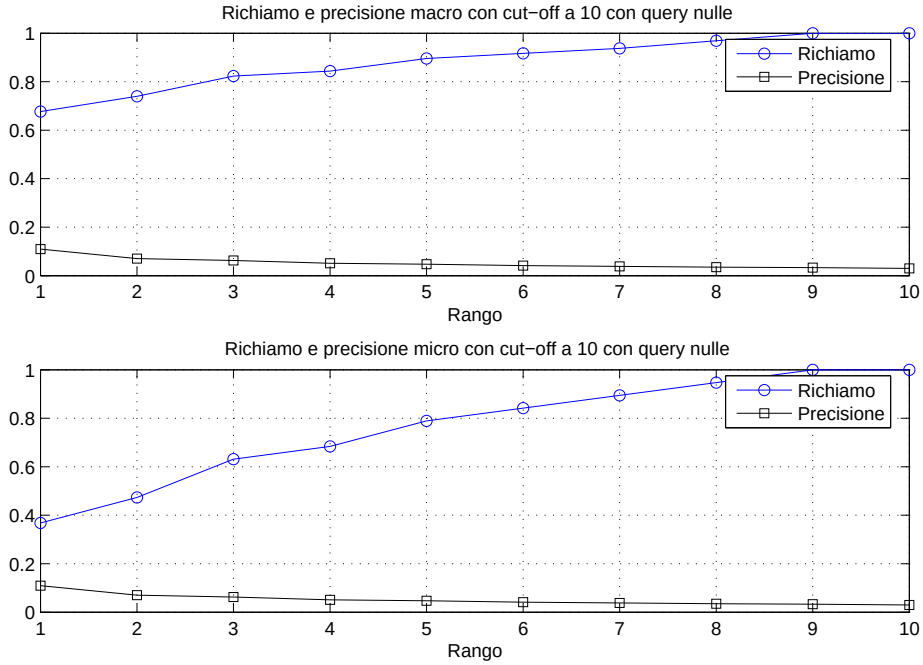

Figure 5: Benchmark con AND-filling

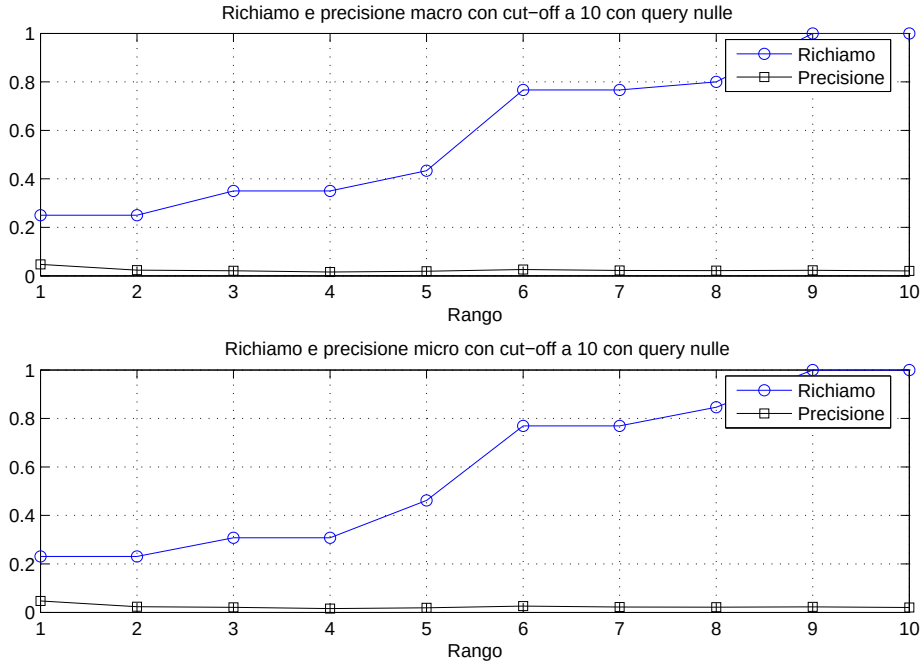

Figure 6: Benchmark con OR-filling

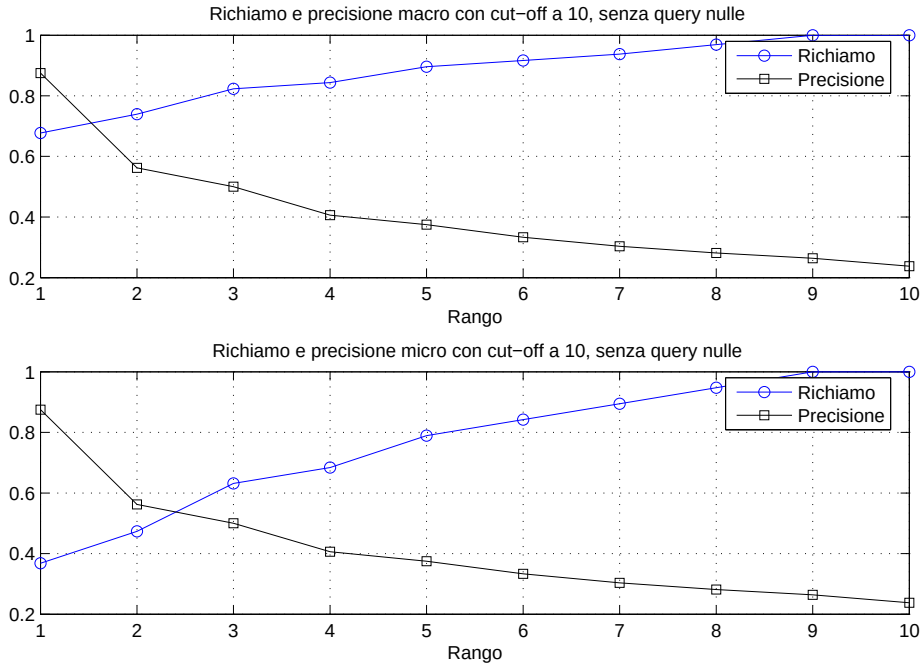

Figure 7: Benchmark con AND-filling

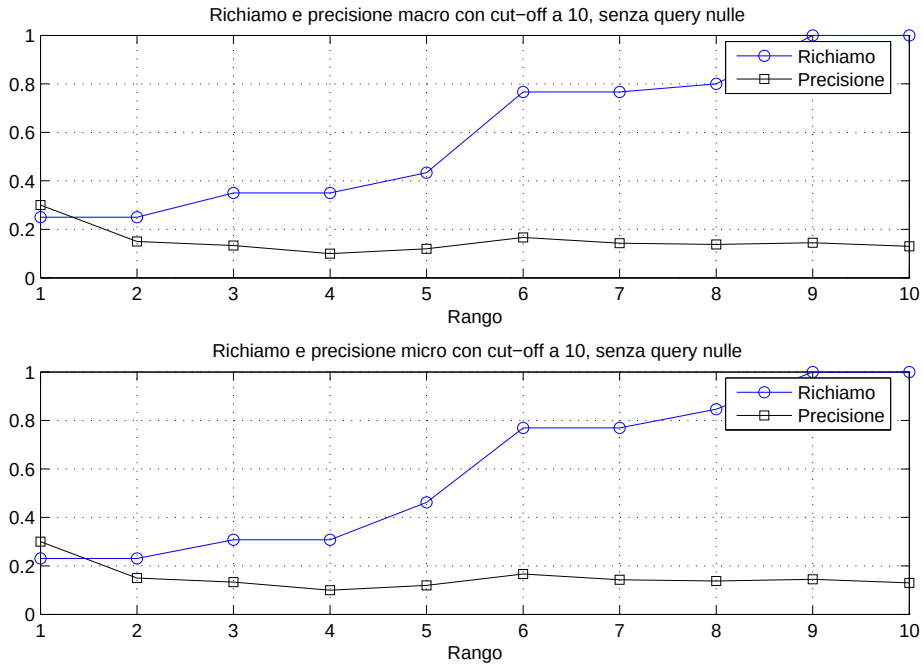

Figure 8: Benchmark con OR-filling

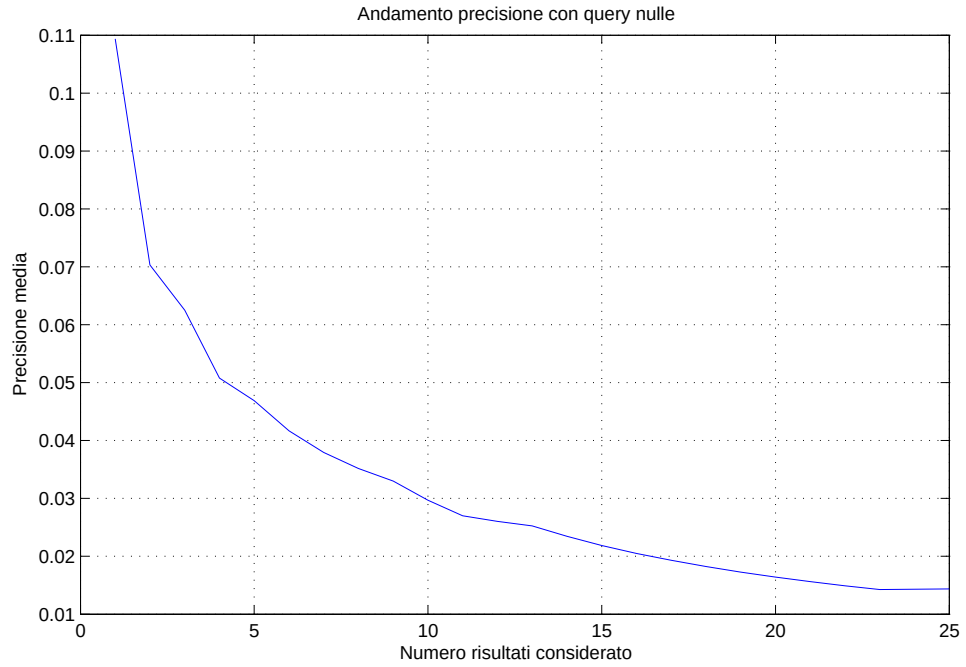

Figure 9: Benchmark con AND-filling

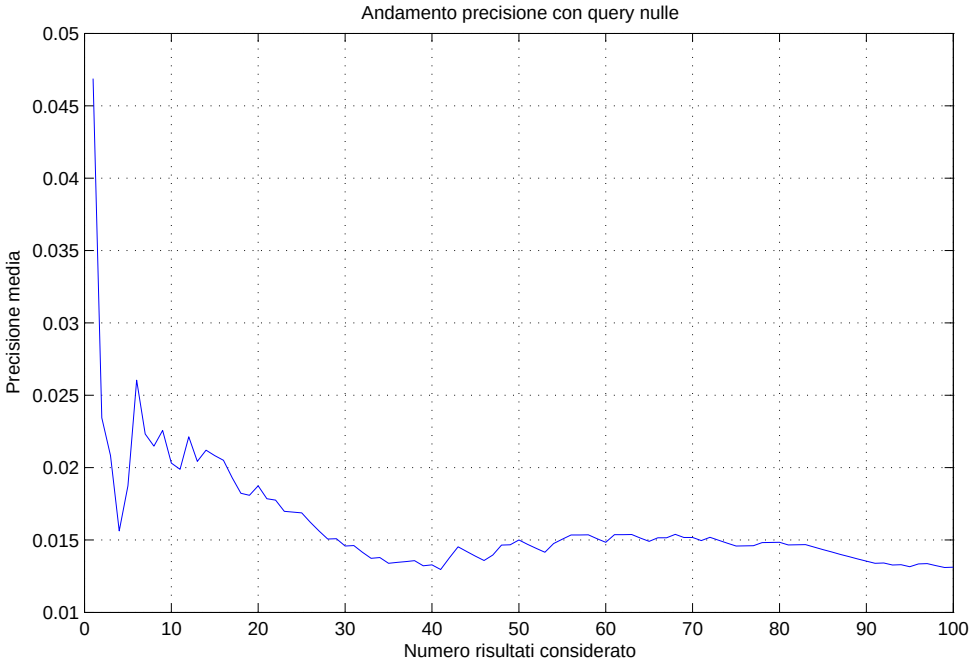

Figure 10: Benchmark con OR-filling

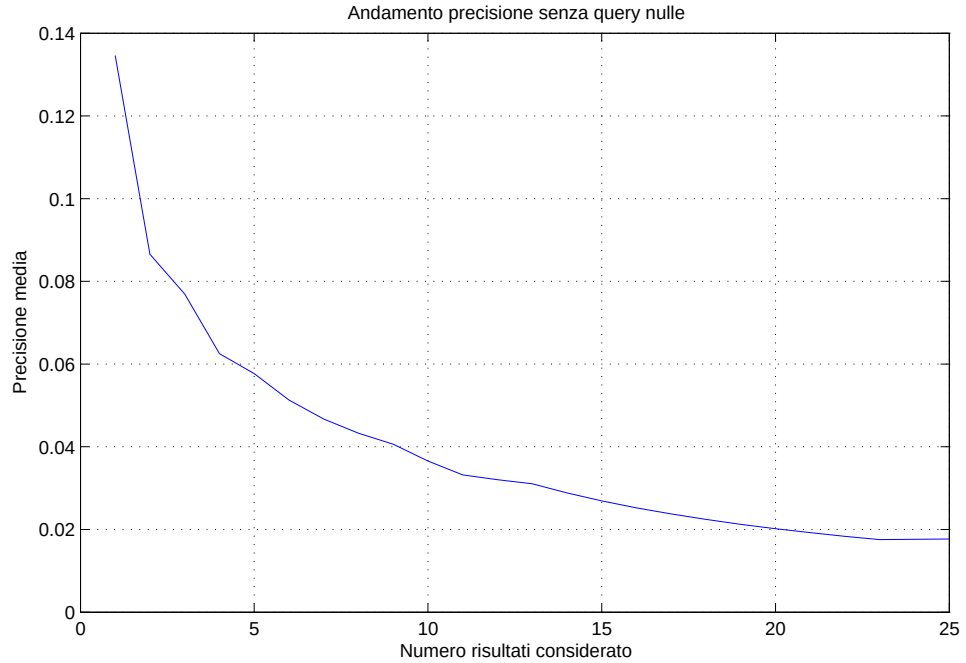

Figure 11: Benchmark con AND-filling

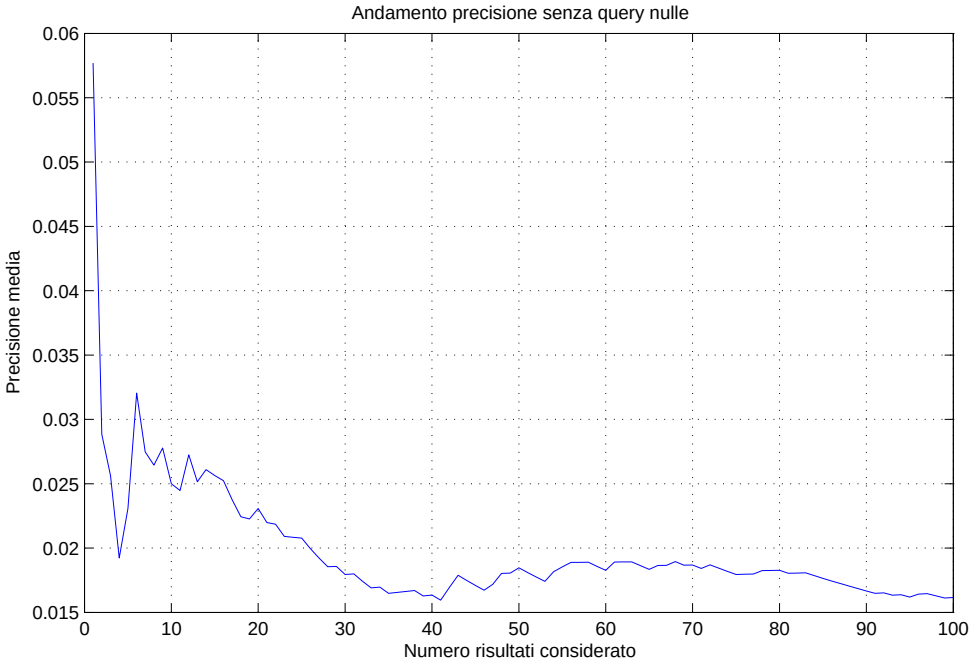

Figure 12: Benchmark con OR-filling

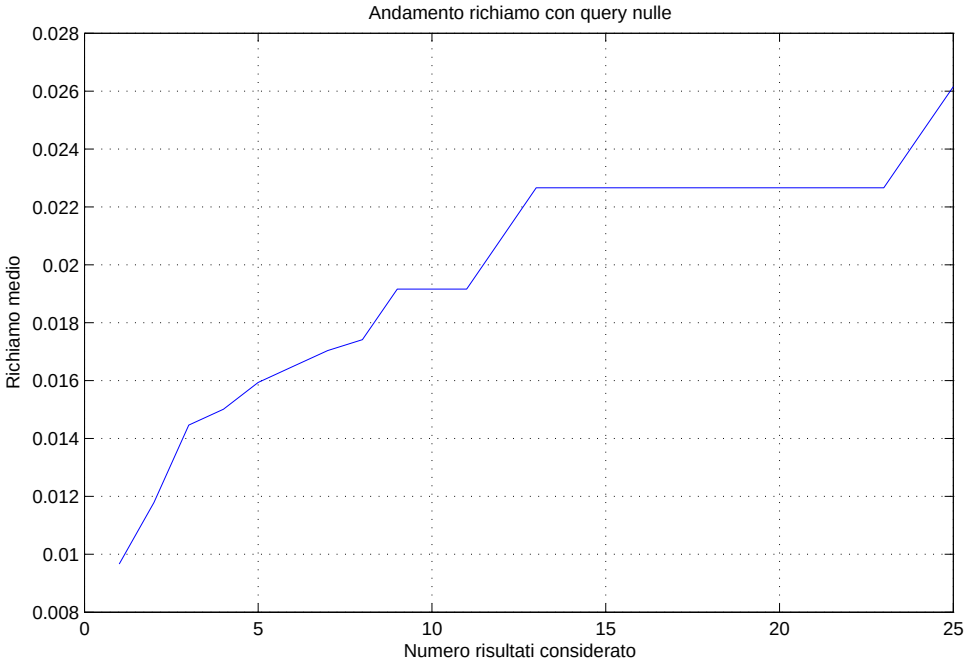

Figure 13: Benchmark con AND-filling

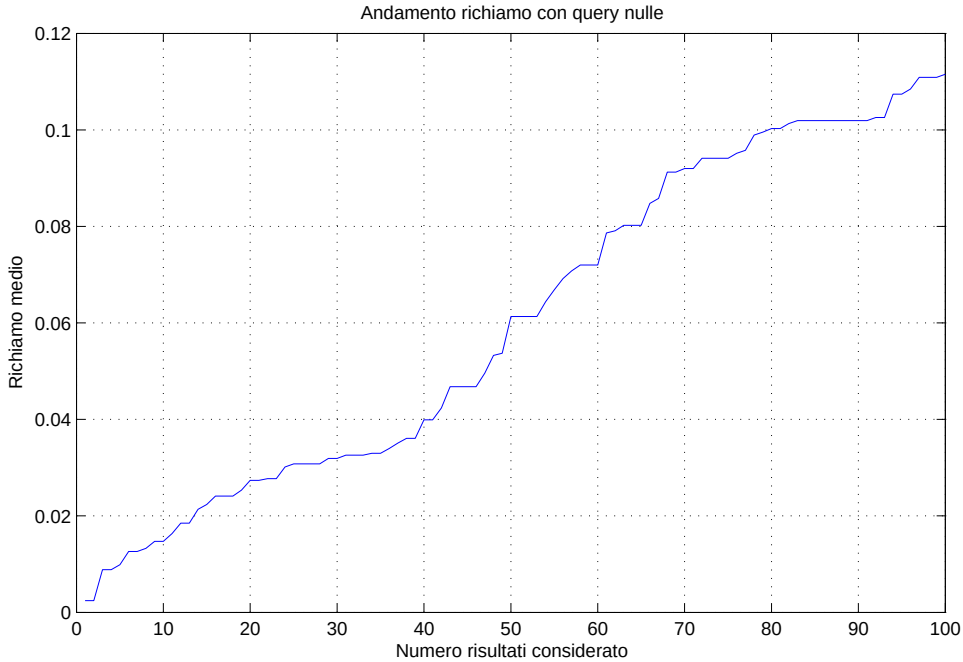

Figure 14: Benchmark con OR-filling

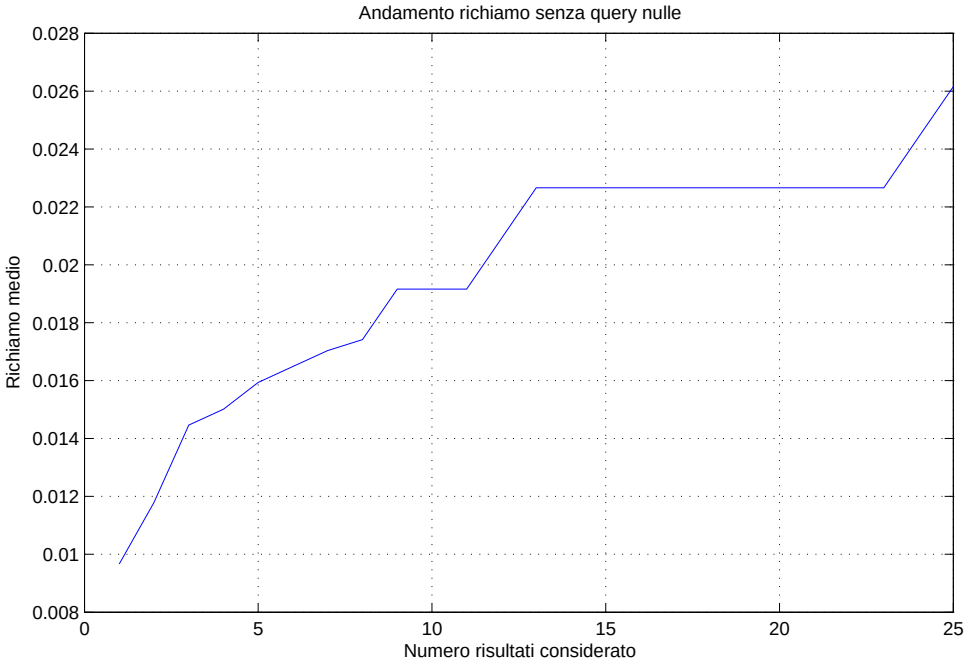

Figure 15: Benchmark con AND-filling

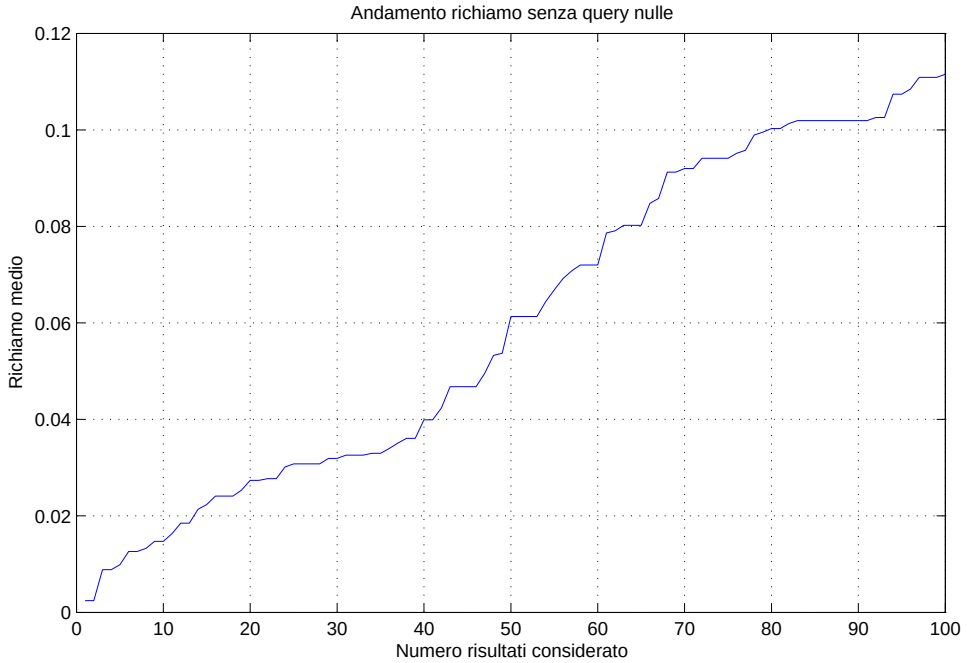

Figure 16: Benchmark con OR-filling

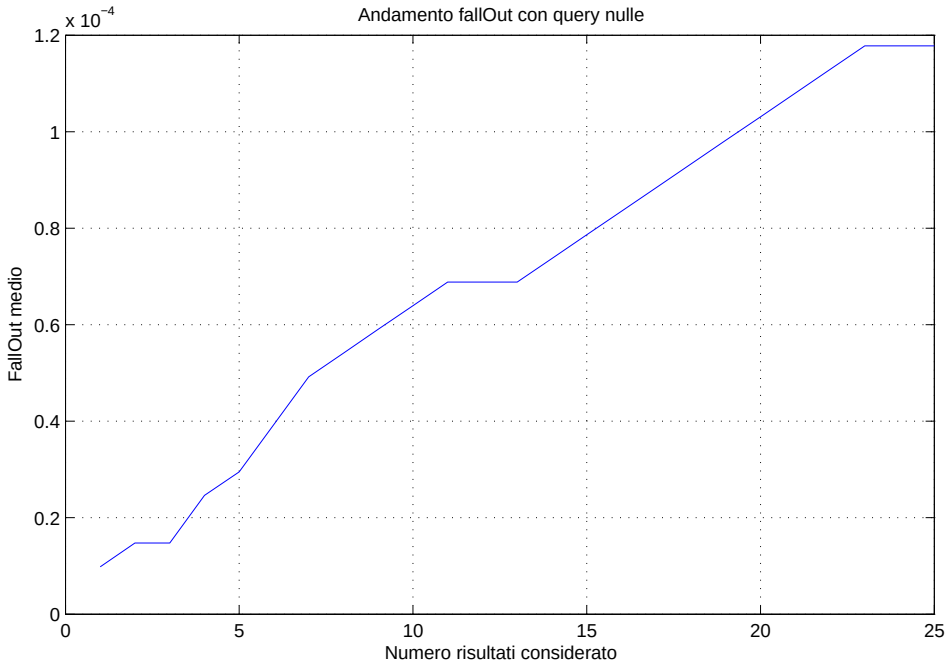

Figure 17: Benchmark con AND-filling

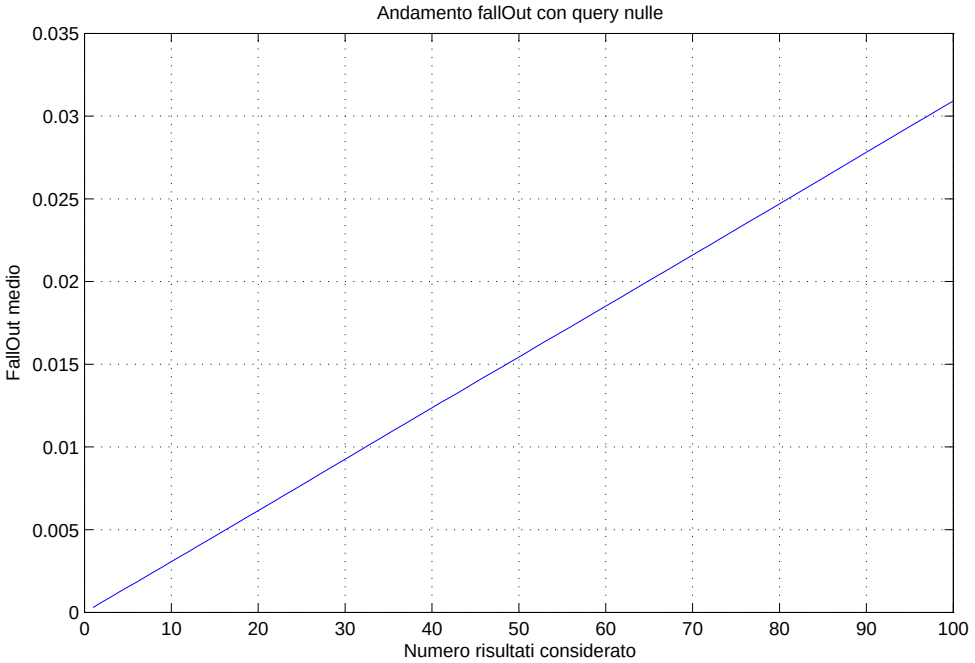

Figure 18: Benchmark con OR-filling

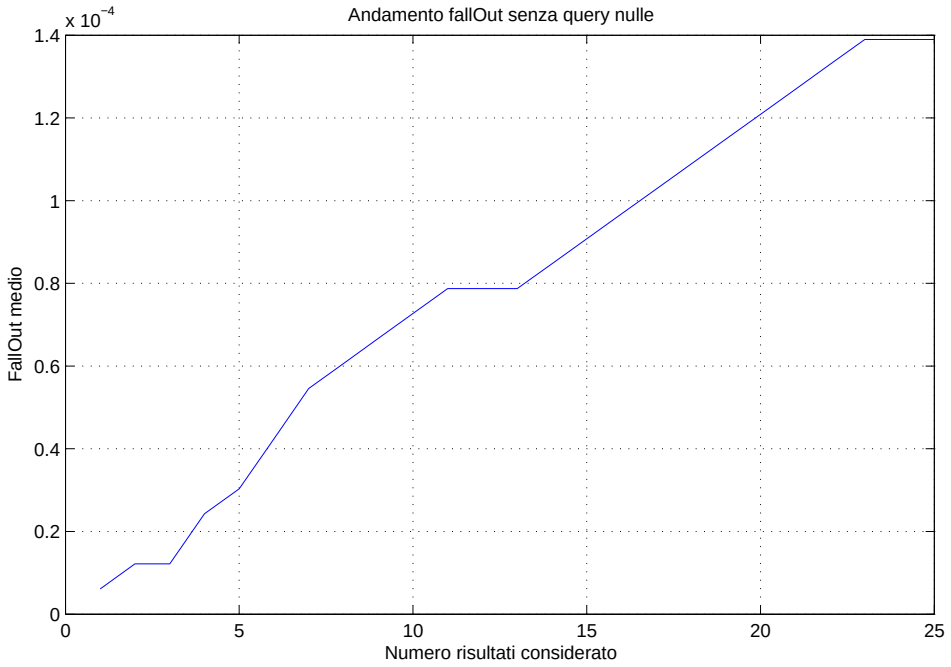

Figure 19: Benchmark con AND-filling

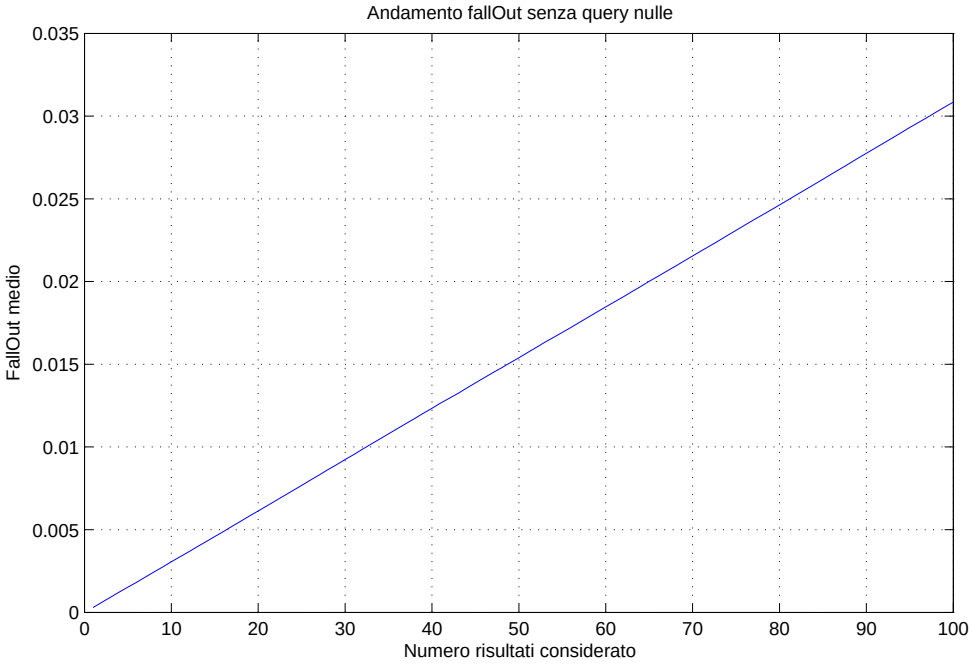

Figure 20: Benchmark con OR-filling