UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Corso di Laurea Magistrale in Informatica

Tesi di Laurea

RIDUZIONE DEL COSTO DELLA VALUTAZIONE DEI SISTEMI DI INFORMATION RETRIEVAL MEDIANTE ALGORITMI GENETICI: IMPLEMENTAZIONE ED ESPERIMENTI

Relatore:
Prof. STEFANO MIZZARO

Laureando:
MICHAEL SOPRANO

Correlatori:
Dott. KEVIN ROITERO
Dott. ANDREA BRUNELLO

ANNO ACCADEMICO 2016-2017

Ai miei genitori ed a mio fratello.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco degli algoritmi

Elenco dei listati

Capitolo 1

Introduzione

Lo scopo di questo capitolo è quello di introdurre brevemente il lavoro svolto in questa tesi. In particolare, nella sezione 1.1 viene descritto l'ambito in cui essa si colloca. Successivamente, nella sezione 1.2 viene descritto l'obiettivo generale da soddisfare e le componenti in cui esso si divide. Infine, nella sezione 1.3 viene riportata una sinossi dei capitoli successivi.

1.1 Ambito

Questa tesi si colloca nell'ambito della disciplina dell'*Information Retrieval*. Tale disciplina può essere descritta con una definizione tratta da Manning et al. [23, p. 1], secondo la quale:

> L'*Information Retrieval* (IR) consiste nel *trovare* materiale (solitamente *documenti*) di natura *non strutturata* (solitamente in una forma *testuale*) al fine di soddisfare un *bisogno informativo* all'interno di una *grande* collezione solitamente archiviata in un server locale o su Internet.

Un *bisogno informativo* viene caratterizzato da tre componenti le quali, in particolare, consistono in un *argomento* (*topic*), un *compito* ed un *contesto*. In altre parole, di cosa parlano i documenti, qual è obiettivo legato alla loro ricerca e tutte le informazioni rimanenti. Alcuni di tali bisogni informativi possono essere caratterizzati dalla necessità di un *alto richiamo*, ossia trovare quanti più documenti possibile relativi all'argomento, mentre altri da quella di avere *alta precisione*, ossia trovare solo alcuni documenti di buona qualità.

Nella figura 1.1 è possibile osservare il modello classico per l'IR. Da tale modello emergono due prospettive, quella dell'utente che utilizza un sistema di IR e quella del

sistema stesso. Per quanto riguarda il punto di vista dell'utente, egli ha un bisogno informativo che cerca di soddisfare sfruttando una collezione di documenti. L'utente stesso, dunque, sottopone un'interrogazione al sistema di IR, il quale restituisce un sottoinsieme di tali documenti. Tale sottoinsieme è formato da quelli più *pertinenti* (ossia con maggior *relevance*) al bisogno informativo espresso nell'interrogazione. Egli, a questo punto, esamina il risultato ottenuto ed, eventualmente, *riformula* il bisogno stesso o la sua espressione in un'interrogazione. Ciascun utente, infatti, potrebbe non sapere esattamente cosa sta cercando oppure come esprimere il tutto. In altre parole, potrebbe avere una percezione errata del bisogno informativo o non riuscire a definirlo sufficientemente bene nell'interrogazione. Per tale motivo, i risultati ottenuti in un'interrogazione precedente possono portare ad una successiva fase di riformulazione del bisogno informativo o dell'interrogazione stessa.

Dal punto di vista del sistema di IR, esso riceve l'interrogazione formulata dall'utente e deve restituire i documenti della collezione che contengono i termini contenuti nell'interrogazione stessa *pesandoli* opportunamente, secondo un qualche criterio. Per migliorare l'efficienza di tale operazione, il sistema costruisce un *indice* dove, per ogni termine contenuto nella collezione, viene indicato quali sono i documenti che lo contengono.

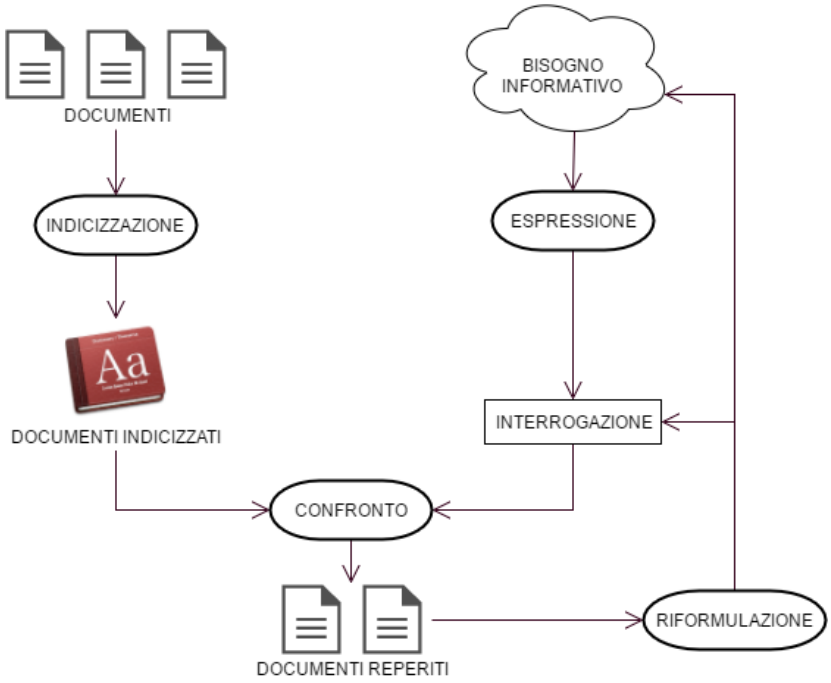


Figura 1.1: Modello classico per l'attività dell'*Information Retrieval*.

I modelli di IR che si possono adottare sono diversi. In generale, quello utilizzato deve specificare come vengono scelti i documenti da reperire e come definirne il *ranking*, il quale viene calcolato assegnando dei pesi ai documenti ed ordinandoli sfruttando i pesi stessi. Per tale motivo, il modello deve specificare anche come calcolare tali valori. Infine, i documenti ordinati in tal modo vengono restituiti all'utente. Uno dei più semplici che si può utilizzare è quello *booleano*. In tale modello, un documento viene reperito se *soddisfa* un'interrogazione, ossia se contiene tutti i termini dell'interrogazione stessa in *AND*, almeno uno di quelli in *OR* e nessuno di quelli in *NOT*, nel senso definito dall'algebra booleana. Per effettuare tale verifica, si può pensare di definire una *funzione di similitudine* $sim(d_j, Q)$ la quale restituisca un valore binario, dove d_j è il documento corrente e Q è l'interrogazione formulata. Se la funzione restituisce un valore pari a 0 secondo le condizioni precedentemente espresse, il documento non viene reperito, mentre se è pari a 1 si. Tale modello, pur essendo molto semplice, soffre di alcune problematiche. In particolare, per l'utente medio diventa complesso esprimere un'interrogazione articolata utilizzando connettivi logici. Inoltre, divide la collezioni in due insiemi (documenti reperiti e non) ma non viene definito nessun ranking.

Alla luce di ciò, è facile immaginare l'esistenza di molti modelli per i sistemi di IR, i quali devono necessariamente affrontare le questioni indicate nella figura 1.2. questa tesi, tuttavia, non vuole descrivere nel dettaglio tali modelli, le tecniche di costruzione di un indice, quelle per riformulare un'interrogazione o quant'altro. Il settore dell'*Information Retrieval* in cui essa si colloca, infatti, è quello della *valutazione* dei sistemi stessi.

Lo scopo del processo di valutazione dei sistemi di IR consiste nel comparare l'efficacia dei sistemi stessi secondo una qualche metodologia che porti ad affermare

Figura 1.2: Tagcloud delle questioni che un modello di IR deve affrontare.

che un dato sistema è migliore di un altro nell'attività di IR. Solitamente, ciò viene svolto testando ciascun sistema mediante un dato insieme di interrogazioni, le quali vengono sottoposte a ciascuno di essi. Ogni sistema, dunque, recupera un insieme di documenti che ritiene pertinenti all'interrogazione ricevuta. Successivamente, tali documenti vengono analizzati da valutatori umani, con il compito di giudicare se essi sono davvero pertinenti all'interrogazione sottoposta o meno. Una volta terminata la fase di assegnazione dei giudizi, vengono calcolate una o più metriche per poter misurare quantitativamente l'efficacia di ciascun sistema rispetto ad ogni interrogazione ed, in tal modo, rispondere alla domanda iniziale.

Nell'ambito del processo di valutazione dei sistemi di IR vi sono tre problematiche principali, le quali vengono descritte nella sezione 2.1. In particolare, tali problematiche consistono nella definizione del concetto di *pertinenza* (*relevance*) di un documento, delle metriche da utilizzare per misurare quantitativamente l'efficacia di ciascun sistema e della metodologia da utilizzare per svolgere il processo di valutazione stesso.

1.2 Obiettivi

L'obiettivo generale di questa tesi si colloca in un preciso ambito di ricerca legato al processo di valutazione dei sistemi di IR il quale mira, in particolare, ad analizzare diversi approcci alla riduzione del *costo* di tale processo. Tale obiettivo consiste nello studiare dettagliatamente uno di tali approcci, il quale viene messo in pratica analizzando una collezione di documenti al fine di investigare riguardo alla miglior scelta possibile di un sottoinsieme ottimo di argomenti (*topic*) da utilizzare per svolgere la valutazione dei sistemi stessi. Tale attività, in particolare, dev'essere svolta prendendo in considerazione due necessità contrastanti, le quali consistono nel riuscire ad individuare un sottoinsieme di argomenti che sia il più piccolo possibile ed, allo stesso tempo, in grado di avere un efficacia migliore (o quantomeno analoga) a quella dell'insieme originale nell'ambito della valutazione stessa.

Diversi ricercatori hanno già svolto un'analisi del processo di riduzione del costo della valutazione dei sistemi di IR mediante la ricerca di sottoinsiemi ottimi di argomenti da utilizzare per svolgere la valutazione stessa. In particolare, Guiver et al. [20] hanno implementato un software chiamato *BestSub* in grado di automatizzare una tale forma di analisi. Il lavoro svolto nell'ambito di questa tesi mira ad implementare una nuova versione di tale software, chiamata *NewBestSub*, la quale sfrutti un rinnovato approccio in grado di cercare sottoinsiemi ottimi di argomenti da uti-

lizzare nella valutazione dei sistemi di IR con una maggiore efficacia ed efficienza cercando, inoltre, di superare le limitazioni che affliggono lo stesso *BestSub*.

Una volta implementato *NewBestSub*, si intende svolgere diversi esperimenti. I primi di essi mirano a riprodurre i risultati ottenuti da tali ricercatori mediante *BestSub* utilizzando lo stesso *NewBestSub*, al fine di certificarne il funzionamento e verificare i miglioramenti ottenuti per quanto riguarda l'efficacia dei risultati e l'efficienza nell'ottenerli. Quelli seguenti hanno lo scopo di generalizzare tali risultati analizzandone alcuni particolari aspetti su un maggior numero di collezioni di documenti rispetto a quanto fatto da Guiver et al. [20]. Infine, gli ultimi esperimenti da svolgere mirano ad espandere gli studi di Guiver et al. [20] al fine di ottenere nuovi risultati.

L'obiettivo generale di questa tesi ed i sotto-obiettivi in cui esso si divide vengono ripresi ed analizzati più dettagliatamente nel capitolo 5 e nelle relative sottosezioni.

1.3 Sinossi

La parte I viene dedicata alla descrizione dello stato dell'arte per quanto riguarda gli argomenti trattati nell'ambito di questa tesi. In particolare, nel capitolo 2 vengono descritti diversi aspetti dell'attività di valutazione dei sistemi di *Information Retrieval*. Inizialmente, viene svolta un'analisi delle questioni principali da affrontare durante tale attività, le quali consistono nella definizione del concetto di *relevance*, nella scelta di metriche adeguate per poter caratterizzare quantitativamente tale concetto e nella scelta di una metodologia adeguata per svolgere l'attività stessa. Successivamente, vengono descritti i due approcci principali alla riduzione del costo di tale processo e vengono discusse le limitazioni di *BestSub*, il software utilizzato da Guiver et al. [20] per investigare il secondo di tali approcci.

Nel capitolo 3 viene presentata la *tecnica meta-euristica* degli *Algoritmi Evolutivi*, il cui utilizzo nell'implementazione della nuova versione di *BestSub*, chiamata *NewBestSub*, ha permesso di superarne le limitazioni. Inizialmente, vi è una descrizione in termini generali di tale tecnica, seguita da quella delle varie entità che caratterizzano ogni algoritmo appartenente ad essa. Successivamente, viene presentato l'algoritmo genetico *NSGA-II*, ossia quello effettivamente utilizzato nell'implementazione di *NewBestSub*. Il capitolo si chiude con la descrizione del nuovo approccio utilizzato per mettere in pratica il processo di riduzione del costo della valutazione dei sistemi di IR studiato da Guiver et al. [20], Robertson [29] e Berto et al. [4].

Il capitolo 4 chiude la parte I e viene dedicato alla descrizione del framework *jMetal*, il quale fornisce l'implementazione di molti algoritmi genetici, tra i quali figura lo stesso *NSGA-II*. Tale framework, dunque, fornisce la basi sulle quali è stato possibile implementare *NewBestSub*. Inizialmente, vengono descritte le motivazioni che hanno portato alla scelta di tale framework e successivamente ne viene presentata l'architettura ed il modo in cui è possibile integrarlo in un software esterno.

La parte II viene dedicata alla descrizione di *NewBestSub* e degli esperimenti svolti sfruttando tale software. In particolare, nel capitolo 5 viene descritto dettagliatamente l'obiettivo generale da soddisfare nell'ambito di questa tesi ed i quattro sotto-obiettivi in cui esso si divide.

Nel capitolo 6 vengono descritte le decisioni prese durante la fase di progettazione architetturale e vi è una descrizione dettaglia di tutte le componenti in cui *NewBestSub* si divide.

Nel capitolo 7 vengono presentate le funzionalità fornite da *NewBestSub* e vi è una descrizione dettagliata di come l'utente può interagire con esso in termini di processo. Inoltre, viene descritta la fase di deployment di *NewBestSub* sul file system ed, infine, vengono descritte le tecnologie utilizzate nella fase d'implementazione.

Nel capitolo 8 vengono descritti gli esperimenti svolti sui risultati ottenuti dalle esecuzioni di tale software sui dataset in input. In particolare, una prima serie di tali esperimenti viene svolta per riprodurre i risultati originali ottenuti con *BestSub*, mentre quelli successivi servono a generalizzare e consolidare tali risultati andando ad analizzare diversi aspetti legati ad essi, per un maggior numero di collezioni di test. Vi è poi un'ulteriore serie di esperimenti che mira ad espandere tali risultati analizzando gli effetti delle metodologie di Soboroff et al. [34] e Mizzaro et al. [25], al fine di ottenere delle strategie pratiche per la selezione a priori di sottoinsiemi di topic da utilizzare per ridurre il costo del processo di valutazione dei sistemi di IR evitando di chiamare in causa valutatori umani. Il capitolo termina, infine, con una descrizione delle tecnologie utilizzate per lo svolgimento di tali esperimenti.

La parte II si chiude con il capitolo 9, nell'ambito del quale vengono tratte le conclusioni di questa tesi e vengono presentati diversi sviluppi futuri. Tali sviluppi futuri, in particolare, possono essere divisi in due categorie. In particolare, quelli appartenenti alla prima categoria riguardano l'implementazione di *NewBestSub* in quanto tale, mentre quelli appartenenti alla seconda consistono in ulteriori esperimenti da svolgere.

Parte I

Stato dell'Arte

Capitolo 2

Valutazione dei Sistemi di Information Retrieval

Lo scopo di questo capitolo è quello di descrivere lo stato dell'arte per quanto riguarda la valutazione dei sistemi di IR e la riduzione del costo di tale processo. In particolare, nella sezione 2.1 vengono descritte le tre questioni principali da affrontare nel corso della valutazione, mentre nella sezione 2.2 vengono descritti i risultati ottenuti dalla ricerca per quanti riguarda l'analisi dei due approcci possibili alla riduzione del costo della valutazione utilizzando meno topic, mentre nella 2.3 viene descritto il software originale chiamato *BestSub* e le limitazioni che lo caratterizzano, il quale viene utilizzato da Guiver et al. [20] per investigare il secondo di tali approcci.

2.1 Problematiche della Valutazione

All'interno del processo di valutazione dei sistemi di IR vi sono tre questioni principali da affrontare, le quali consistono nella definizione del concetto di *relevance*, nella scelta delle *metriche* da utilizzare per la misura della relevance stessa e nella definizione di una *metodologia* per svolgere tale attività di valutazione.

La descrizione del concetto di relevance è tratta da Mizzaro [24, p. 1-11]. Uno dei primi aspetti da prendere in considerazione è relativo al fatto che esiste più di una tipologia di *relevance*. Un modo per classificare tali tipologie consiste nel definire uno spazio composto da quattro dimensioni, dove ciascuna di esse viene caratterizzata da un insieme ordinato di entità. Ogni *relevance*, dunque, è un punto nello spazio definito da tali dimensioni.

2.1.1 Relevance

La prima dimensione della *relevance* è la tipologia di *risorsa informativa* per la quale definire la relevance stessa. Lungo tale dimensione è possibile distinguere tre entità diverse. La prima di esse è il *Documento*, l'entità fisica che l'utente di un sistema di IR ottiene dopo una ricerca. Vi è poi il *Surrogato*, una rappresentazione del documento composta da uno o più metadati quali titolo, nome dell'autore, elenco delle parole chiave ed altro. Infine, vi è l'*Informazione*, l'entità non fisica che l'utente riceve o crea quando legge un documento. In tal modo, è possibile definire l'insieme delle *risorse informative* 2.1, il quale viene caratterizzato dall'ordinamento 2.2.

$$InfRes = \{Surrogato, Documento, Informazione\} \tag{2.1}$$

$$Surrogato < Documento < Informazione \tag{2.2}$$

La seconda dimensione della *relevance* si basa sull'assunzione che l'utente svolge una ricerca quando si trova in una "situazione problematica", la quale può essere risolta con successo solamente ottenendo delle informazioni. In altre parole, l'utente ha un *bisogno informativo* da soddisfare (*Real Information Need*, o *RIN*). Percependo il *RIN* l'utente stesso costruisce il *bisogno informativo percepito*, ossia una rappresentazione mentale del bisogno stesso (*Perceived Information Need*, o *PIN*). Successivamente, il *PIN* viene espresso in una *richiesta* (*Expressed Information Need*, o *EIN*), la quale viene formalizzata in un *interrogazione* da sottoporre al sistema (*Formalized Information Need*, o *FIN*). In tal modo, è possibile definire l'insieme di *rappresentazioni del problema dell'utente* 2.3, il quale viene caratterizzato dall'ordinamento 2.4. In realtà, la scelta degli elementi dell'insieme *Repr* non è fondamentale poiché l'aspetto chiave consiste nel comprendere che esistono *diverse* rappresentazioni del problema dell'utente.

$$Repr = \{RIN, PIN, EIN, FIN\} \tag{2.3}$$

$$FIN < EIN < PIN < RIN \tag{2.4}$$

A questo punto è già possibile individuare alcune delle varie *relevance*. Per esempio, vi è quella dell'informazione ottenuta dall'utente rispetto al *RIN*, quella di un documento rispetto ad un'interrogazione, ecc. L'ordinamento definito negli insiemi che rappresentano le prime due dimensioni induce un ordinamento parziale delle relevance nello spazio formato dalle dimensioni stesse, visibile nella figura 2.1. Il senso di tale ordinamento è che quelle nella "sezione superiore" dello spazio sono

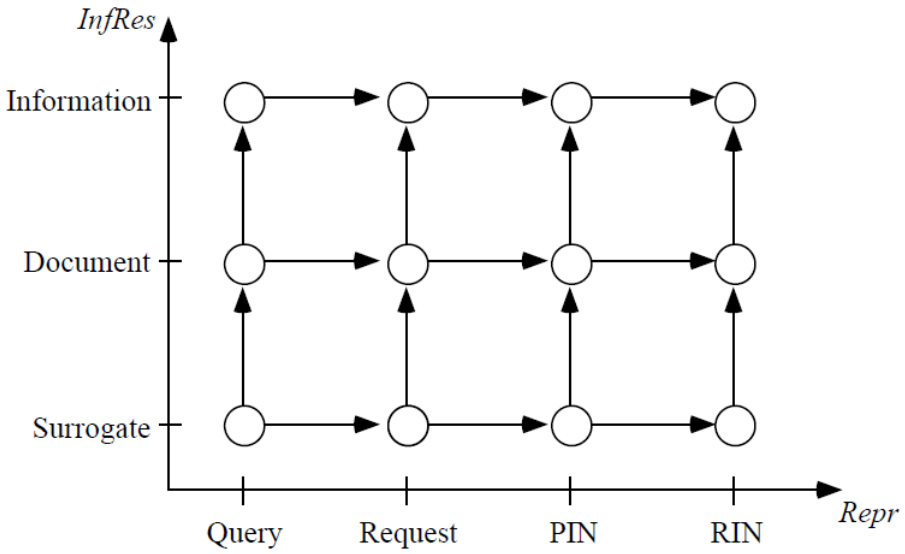


Figura 2.1: La *relevance* come punto in uno spazio bidimensionale, tratta da Mizzaro [24, p. 7].

più vicine all'utente e più difficili da misurare, mentre quelle presenti nella "sezione inferiore" sono più vicine al sistema e, dunque, più operative e di facile misura.

La terza dimensione della *relevance* è data dal *tempo*. Un documento (un surrogato, un'informazione) può essere pertinente ad un'interrogazione (una richiesta, un *PIN*, un *RIN*) e non esserlo più successivamente. Tra il sistema di IR e l'utente avvengono interazioni dinamiche e continue, perciò la relevance stessa non può essere statica. L'utente, dunque, ha un *RIN* iniziale $t(rin_0)$ che percepisce, ottenendo il primo *PIN* all'istante $t(pin_0)$. Successivamente il *PIN* viene espresso nella prima richiesta ottenendo $t(ein_0)$, la quale viene formalizzata nella prima interrogazione $t(fin_0)$. La richiesta, l'interrogazione ed il bisogno percepito possono subire delle revisioni fino a raggiungere la configurazione finale $t(pin_p)$, $t(ein_m)$ e $t(fin_n)$, dove p, m ed n sono istanti temporali successivi. Tali considerazioni portano a definire l'insieme degli *istanti temporali* 2.5 che vanno dalla nascita del *RIN* dell'utente alla sua soddisfazione. L'ordinamento dell'insieme è, dunque, quello temporale, ottenuto leggendo gli elementi dell'insieme da sinistra verso destra.

$$\begin{aligned} Time = \{&t(rin_0), t(pin_0), t(ein_0), t(fin_0), t(ein_1), t(fin_1), \\ &\ldots, t(pin_p), \ldots, t(ein_n), \ldots, t(fin_n), t(f)\} \end{aligned} \tag{2.5}$$

La quarta ed ultima dimensione della *relevance* è quella data dalle *componenti*. Tali componenti sono il *topic* (*To*), ossia ciò di cui parla il documento reperito, il *task*

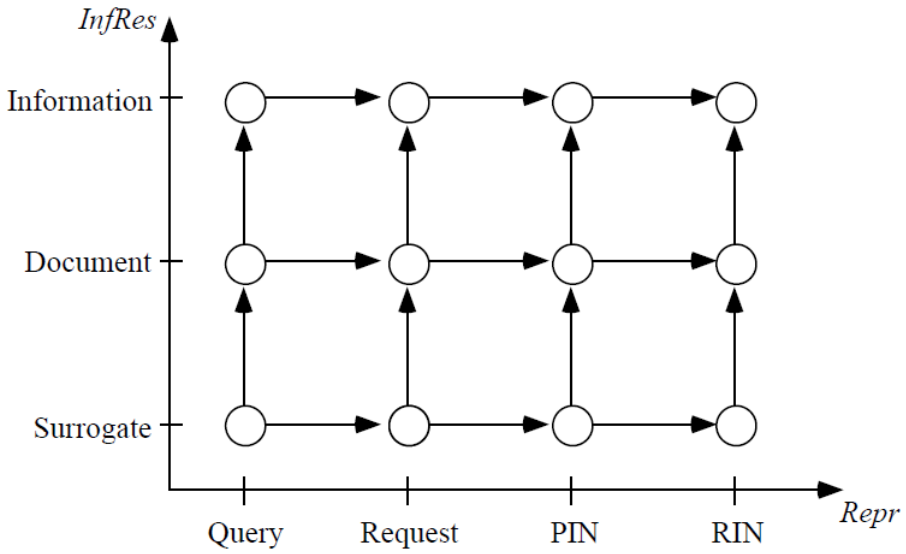


Figura 2.2: L'ordinamento parziale dell'insieme *Comp*, tratto da Mizzaro [24, p. 9].

(*Ta*), ossia lo scopo per il quale il documento stesso è necessario, ed il *contesto* (*Co*), ossia ciò che non riguarda le prime due componenti ma che finisce per influenzare come l'utente svolge la ricerca e come vengono valutati i risultati ottenuti. Per tale motivo, un documento (un surrogato, un'informazione) può essere pertinente rispetto al *RIN* (*PIN*, *EIN*, *FIN*) per qualsiasi sottoinsieme delle tre componenti. In tal modo, è possibile definire l'insieme delle *componenti* 2.6, dove $\mathcal{P}$ è l'insieme delle parti. A differenza di quelli precedenti, in questo insieme l'ordinamento è solamente parziale e viene definito dalla condizione 2.7. Una rappresentazione di tale ordinamento è visibile nella figura 2.2.

$$Comp = \mathcal{P}(To, Ta, Co) - \emptyset \tag{2.6}$$

$$\forall x, y \in Comp\ (x < y \iff x \subset y) \tag{2.7}$$

Ricapitolando, ogni *relevance* può essere vista come un punto in uno spazio quadridimensionale, dove i valori di ogni dimensione sono dati dagli elementi degli insiemi *InfRes*, *Repr*, *Time* e *Comp*. In tal modo, è possibile definire *l'insieme parzialmente ordinato delle relevance* mediante il prodotto cartesiano degli insiemi precedenti. La condizione che definisce l'ordinamento parziale di tale insieme viene omessa, ma è possibile vederne una rappresentazione di tale ordinamento nella figura 2.8 la quale, tuttavia, non considera l'ordinamento temporale.

$$Relevances = InfRes \times Repr \times Time \times Comp \tag{2.8}$$

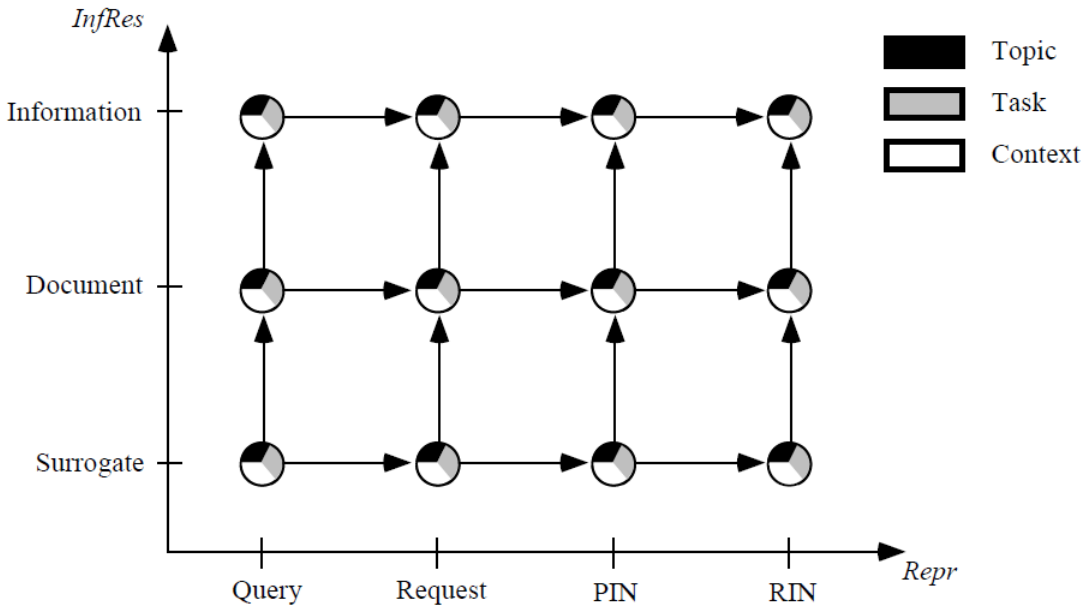


Figura 2.3: Rappresentazione delle diverse tipologie di *relevance*, tratta da Mizzaro [24, p. 10].

Ogni *relevance*, dunque, viene rappresentata da una quadrupla di elementi. Per esempio, $rel(S, Q, t(q), \{To\})$ è quella di un surrogato per un'interrogazione all'istante temporale in cui essa viene svolta per quanto riguarda l'argomento, mentre $rel(I, RIN, t(f), \{To, Ta, Co\})$ è quella di un'informazione per il bisogno informativo reale dell'utente rispetto a tutte le componenti, ossia quella a cui l'utente stesso aspira. Sulla base di tale formalizzazione, un sistema di IR può essere più o meno efficace per *relevance* diverse e tale aspetto va preso in considerazione durante la fase di valutazione del sistema stesso.

Un *giudizio di relevance*, dunque, può essere definito come l'assegnamento di un valore per *una* delle varie relevance svolto da un *valutatore* ad un determinato *istante temporale*. Si possono individuare diverse tipologie di tali giudizi, le quali sono classificabili lungo cinque dimensioni caratterizzate da altrettanti insiemi ordinati, similmente a quanto avviene per la relevance in quanto tale (Mizzaro [24, p. 11]).

2.1.2 Metriche

La seconda problematica da affrontare durante la fase di valutazione è legata alla definizione di *metriche* da utilizzare per misurare quantitativamente le relevance da valutare. Nel corso degli anni, sono state definite molte metriche diverse ed una

buona parte di esse fanno affidamento sulle formule classiche 2.9, le quali permettono di calcolare i valori di *Precisione* (*Precision*) e *Recupero* (*Recall*) di un sistema di IR. In particolare, con *pert* si fa riferimento all'insieme dei documenti pertinenti, mentre con *rec* a quello dei documenti recuperati.

$$P = |pert\&rec|/|rec| \quad \text{e} \quad R = |pert\&rec|/|pert| \tag{2.9}$$

Il problema più rilevante di tali formule consiste nel fatto che i documenti recuperati da un sistema di IR vengono ordinati al fine di stabilire un ranking ed il sistema stesso dovrebbe essere premiato qualora recuperasse i documenti pertinenti tra le prime posizioni del ranking definito. I valori di P ed R non prendono in considerazione tale aspetto, perciò risultano necessarie metriche più evolute di esse. Ciò, in particolare, è dovuto a ragioni storiche; i primi sistemi di IR costruiti, infatti, erano basati sul modello booleano il quale, come si è visto, si limita semplicemente a dividere l'insieme dei documenti nei sottoinsiemi di quelli recuperati e di quelli pertinenti. Un ulteriore problematica consiste nel fatto che i valori di P ed R fanno riferimento ad una singola interrogazione. Un documento, tuttavia, potrebbe essere pertinente ad una prima interrogazione e non esserlo più per quella seguente. Diventa necessario, dunque, prendere in considerazione anche tale aspetto nel processo di valutazione di un sistema di IR.

Una metrica che permette di prendere in considerazione la pertinenza di un documento ad una singola interrogazione e la sua posizione del ranking di quelli reperiti è la *Precisione Media* (*Average Precision*), definita in Baeza-Yates et al. [2, p. 140]. Per un esempio di calcolo di tale valore, si supponga di aver recuperato dieci documenti per un'interrogazione e di aver ottenuto il ranking descritto nella tabella 2.1. Per ogni posizione vengono calcolati i valori di P ed R e viene stabilito se il documento è pertinente o meno. Il valore di AP è la media di quelli di P per le posizioni del ranking per le quali il documento recuperato risulta pertinente all'interrogazione stessa. Applicando tale formula al ranking descritto nella tabella 2.1, dunque, si ha che:

$$AP = (1 + 1 + 0.75 + 0.57)/4 = 0.83 \tag{2.10}$$

Una variante di tale metrica consiste nell'utilizzare il logaritmo del valore di AP calcolato ($LogAP$). Per valutare un sistema di IR su più interrogazioni viene calcolata la *Mean Average Precision* (MAP), dove tale valore corrisponde alla media delle AP calcolate per ciascuna di tali interrogazioni.

La forma di *relevance* che AP permette di ottenere è binaria, con un reperimento basato sul ranking. In altre parole, un documento può essere pertinente o meno,

Pos.	Pert?	R	P
1	1	.25	1
2	1	.50	1
3	0	.50	.67
4	1	.75	.75
5	0	.75	.60
6	0	.75	.50
7	1	1	.57
8	0	1	.50
9	0	1	.44
10	0	1	.40

Tabella 2.1: Dati per un esempio di calcolo della *Average Precision*.

senza vie di mezzo, e la sua posizione nel ranking di quelli reperiti è importante. In alcune situazioni può risultare interessante permettere al valutatore umano di esprimere un giudizio *categoriale* che prenda comunque in considerazione anche la posizione del documento nel ranking. Tale giudizio categoriale permette al valutatore umano di esprimere il fatto che un dato documento è più o meno pertinente di altri documenti a loro volta pertinenti. Una metrica che permette di ottenere tutto ciò é il *Discounted Cumulative Gain* (DCG). Secondo tale metrica, dunque, è possibile definire $\{O, \ldots, N-1\}$ gradi di relevance, con l'assunzione che i documenti più pertinenti debbano essere recuperati nelle prime posizioni del ranking poiché, intuitivamente, l'utente ha un "guadagno" maggiore in termini di soddisfazione del proprio bisogno informativo da essi. Il DCG, dunque, misura il "guadagno" che un documento fornisce all'utente "scontandolo" del logaritmo del rango. Per valutare un sistema di IR su più interrogazioni, è sufficiente calcolare la media dei valori di DCG ottenuti per ciascuna di esse. Tale metrica, tuttavia, è caratterizzata da una particolare problematica, la quale consiste nel fatto che interrogazioni con molti documenti pertinenti hanno un peso maggiore di quelle che che hanno pochi. In altre parole, interrogazioni con "molta relevance" hanno "maggior guadagno". Per evitare tutto ciò viene definita una normalizzazione del DCG chiamata *Normalized*

Discounted Cumulative Gain (*NDCG*). Tale normalizzazione consiste, intuitivamente, nel dividere il valore di *DCG* per il valore "ideale" che il rango del documento dovrebbe avere. Successivamente, viene calcolata la media per i valori sulla stessa scala, dove ciascuno di essi ha 1 come massimo. Il risultato di tale operazione è il valore finale di *NDCG*.

In definitiva, ognuna di tali metriche ha aspetti positivi e negativi. Per tale motivo, è necessario analizzare il problema e scegliere quella più adatta agli scopi del processo di valutazione.

2.1.3 Metodologie

La terza problematica da affrontare nell'attività di valutazione di un sistema di IR consiste nella scelta di una *metodologia* per svolgere tale attività. Vi sono due approcci principali, i quali consistono nell'analisi di una *collezione di test* che funge da *benchmark* per l'efficacia del sistema stesso oppure nello *studio degli utenti* durante l'uso del sistema. Entrambe le metodologie hanno aspetti positivi e negativi e si completano a vicenda. Ad ogni modo, in questa tesi ci si concentra principalmente sull'analisi delle collezioni di test.

Una *collezione di test* (*test collection*) è costituita da un insieme di *documenti* e da un insieme di *richieste* (*topics*). Ciascun *topic* è una descrizione di un determinato bisogno informativo ed i documenti pertinenti al bisogno stesso tra quelli presenti nella collezione sono noti a priori, poiché vi sono valutatori umani che assegnano un *giudizio di relevance* (*qrel*) a ciascun documento della collezione rispetto a ciascun topic. Per effettuare la valutazione, diversi sistemi di IR vengono comparati sulla collezione stessa, effettuando una ricerca per ciascun topic e misurandone l'efficacia secondo una o più metriche. Se la collezione è molto grande, tuttavia, può essere complesso riuscire a trovare tutti i documenti pertinenti. Per superare tale problematica, viene adottato il metodo del *pooling*. Tale metodo consiste nel considerare solamente i primi N documenti recuperati da ciascun sistema, i quali costituiscono il *pool* da valutare. Successivamente, solo i membri di tale pool vengono effettivamente valutati ed i rimanenti vengono considerati automaticamente come non pertinenti. La speranza di fondo su cui si basa tale metodo consiste nell'assumere che un documento pertinente venga recuperato da almeno un sistema, nelle prime N posizioni del ranking.

Nel mondo vi sono diversi eventi dove vengono testati sistemi di IR sfruttando

collezioni di test. Uno dei più noti è la *Text REtrieval Conference* (*TREC*)[1], sponsorizzata dal *National Institute of Standards and Technology* (*NIST*). Tale evento consiste in una serie di workshop che si svolgono annualmente, il cui scopo è stimolare la ricerca sulle tematiche dell'IR fornendo grandi collezioni di test e l'infrastruttura necessaria per svolgere l'attività di valutazione dei sistemi partecipanti.

Gli obiettivi di ricerca di *TREC* vengono chiamati *task*. Quello chiamato "Ad Hoc" è il task classico, dedicato alla valutazione dei sistemi di IR che recuperano documenti definendone un ranking da una collezione di test costruita appositamente. In particolare, tale collezione è stata espansa di anno in anno utilizzando, inizialmente, articoli scientifici, documenti governativi ed altro, per poi sfruttare anche documenti recuperati da specifiche sessioni di crawling di parti del Web o di specifici domini. Per quanto riguarda i *topic*, ogni anno ne vengono aggiunti circa cinquanta ed essi includono anche le informazioni relative a come il valutatore umano dovrebbe decidere se un documento è pertinente o meno al topic stesso. La descrizione che viene svolta nel resto di questa sezione è tratta da Vorhees et al. [38].

Per partecipare ad un workshop di *TREC* con un sistema di IR è possibile preparare il sistema stesso indicizzando le collezioni degli eventi precedenti. Inoltre, è possibile ottenere anche i topic negli anni precedenti al fine di regolare il sistema stesso svolgendo dei test. Una volta completata la fase preparatoria è possibile richiedere i topic per l'anno corrente; una volta ottenuti, viene eseguita una ricerca per ciascuno di esso mediante il sistema da analizzare. I risultati ottenuti vengono inviati ai valutatori umani in un dato formato, che consiste nel ranking dei primi mille documenti recuperati per ciascun topic.

Per svolgere l'attività di valutazione, vengono ottenuti i mille documenti inviati da ogni partecipante per ciascuno dei cinquanta topic (*run*). Successivamente, viene svolta un'attività di *pooling* mediante la quale vengono scelti i primi cento documenti di ciascuna *run*, dove quelli rimanenti vengono considerati automaticamente come non pertinenti. A questo punto, i valutatori umani decidono sulla pertinenza di ciascuno dei documenti del *pool* al topic corrente. La decisione che viene espressa dal valutatore è binaria e vengono creati dei file contenenti tali *giudizi di relevance* (*qrels*). Infine, il software ufficiale di *TREC* chiamato `trec_eval` viene eseguito sui risultati ottenuti al fine di calcolare diverse metriche per le performance dei sistemi.

Nell'ambito del task scelto vengono proposti diversi *tracks* , che consistono nello svolgere l'attività di esso utilizzando collezioni e metriche diverse. Per il task "Ad Hoc", ad esempio, vi sono i track chiamati *Web*, *Terabyte*, *Robust* e *Million Query*

[1] `http://trec.nist.gov/`

(Hawking et al. [21], Clarke et al. [7], Vorhees [37] e Allan et al. [1]). Il primo è caratterizzato da una collezione costituita da un sottoinsieme di documenti recuperato dal Web, mentre il secondo si concentra su collezioni molto grandi, per le quali è estremamente complesso ottenere tutti i documenti pertinenti, dove nemmeno il pooling risulta efficace. Il terzo si concentra su topic ritenuti "difficili" da soddisfare, mentre il quarto ed ultimo track è stato definito alcuni anni dopo *Terabyte* ed ha come scopo l'analisi di collezioni ancora più grandi e di diverse questioni metodologiche relative alla valutazione dei risultati delle interrogazioni da parte dei valutatori umani.

Il task "Ad Hoc" è attualmente disattivo, poiché è difficile ottenere ulteriori miglioramenti nell'efficacia dei sistemi per quanto riguarda l'IR classica. Tuttavia, rimane una valida piattaforma di test per valutarne quelli di nuova implementazione.

2.2 Riduzione del Costo della Valutazione

Per ridurre il numero di topic utilizzati nella valutazione dell'efficacia di un sistema di IR vi sono due approcci principali, i quali consistono nella scelta di un sottoinsieme di topic di cardinalità inferiore attraverso una forma di campionamento casuale o nell'investigare riguardo alla miglior scelta possibile di un sottoinsieme ottimo di topic.

Il primo approccio è stato analizzato da molti ricercatori i quali, tuttavia, sono giunti a conclusioni diverse. In particolare, Sparck Jones et al. [35, p. 63] analizzano il numero di topic utilizzati nella valutazione mediante collezioni di test e concludono che ≪250 (topics) sono generalmente accettabili ed a volte ne sono necessari 1000≫. Diversi anni dopo, Zobel [42], focalizzandosi sulla profondità dei pool di documenti, conclude che un insieme da 25 topic permette di prevedere la valutazione dell'efficacia dei sistemi di IR su un diverso insieme da 25 topic, mentre Buckley et al. [6, p. 39] affermano che ≪25 topic risultano appena sufficienti per un esperimento ma 50 topic sono stabili≫. Più recentemente, Webber et al. [40] e Sakai [32, p. 256] sfruttano strumenti statistici ed il primo ottiene che ≪un esperimento dovrebbe contenere almeno 150 topic≫, mentre il secondo afferma che ≪poiché misure per l'efficacia diverse possono avere varianze notevolmente differenti nell'ambito di ciascun sistema, esse richiedono che gli insiemi di topic utilizzati abbiano dimensioni sostanzialmente diverse, al di sotto degli stessi requisiti statistici≫. Altri ricercatori, inoltre, mostrano l'esistenza di forti interazioni tra sistemi e topic. Tali lavori includono l'analisi ANOVA di Banks et al. [3], i risultati di Mizzaro et al. [25] e Roitero et al. [31],

	t_1	$\cdots$	t_n	MAP
s_1	$\text{AP}(s_1, t_1)$	$\cdots$	$\text{AP}(s_1, t_n)$	$\text{MAP}(s_1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
s_m	$\text{AP}(s_m, t_1)$	$\cdots$	$\text{AP}(s_m, t_n)$	$\text{MAP}(s_m)$

Figura 2.4: Matrice con vettori di AP e valori di MAP per n topic e m sistemi, tratta da Guiver et al. [20].

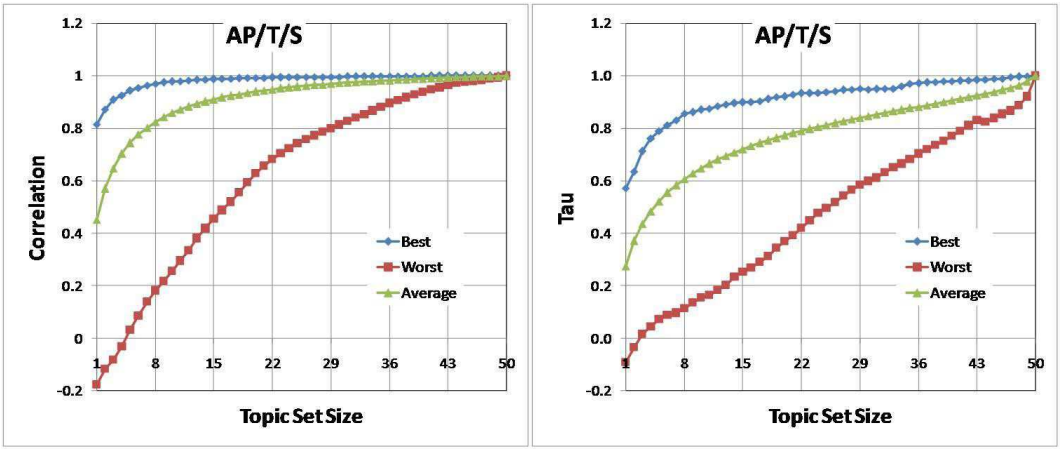
i quali portano alla luce tali interazioni utilizzando tecniche di analisi delle reti, e quelli di Kleinberg [22].

Per quanto riguarda il secondo approccio, i tre contributi principali sono quelli di Guiver et al. [20], Robertson [29] e Berto et al. [4] e poiché uno degli obiettivi di questa tesi consiste nella riproduzione dei risultati ottenuti nell'ambito di tali contributi, essi vengono descritti più dettagliatamente nel seguito.

Guiver et al. [20] propongono un'analisi teorica sulla riduzione del numero di topic da utilizzare durante la fase di valutazione dei sistemi di IR la quale parte dai risultati ottenuti da *TREC*, disponibili nel formato rappresentato nella figura 2.4. In particolare, il processo viene descritto in Guiver et al. [20, pp. 4–21] come segue:

> Iniziamo da un insieme di n topics ($n = 50$ o 25 nell'esperimento che segue). A questo punto consideriamo, per qualsiasi $c \in \{1, \ldots, n\}$ e per qualsiasi sottoinsieme di topic di cardinalità c, il valore di *MAP* calcolato per ogni sistema solamente su tale sottoinsieme di topic. In altre parole, calcoliamo la media di un sottoinsieme di dimensione c delle n colonne della tabella visibile nella figura 2.4. Per ognuno di tali sottoinsiemi, inoltre, calcoliamo la correlazione dei rispettivi valori di *MAP* con quelli dell'insieme originale di colonne (topic). Tale valore di correlazione misura quanto efficacemente il sottoinsieme selezionato prevede le performance di diversi sistemi sull'insieme originale. A questo punto, per ogni cardinalità c selezioniamo il miglior sottoinsieme di topic (*Best*), ossia quello con la maggior correlazione con quello originale. Inoltre, selezioniamo anche quello con la peggior correlazione (*Worst*) ed infine calcoliamo una correlazione media per tutti i sottoinsiemi di topic di dimensione c (Average).

Il risultato di tale processo è visibile nelle figure 2.5, le quali mostrano tre valori di correlazione (asse y) per ciascuna cardinalità (asse x). In particolare, quelli



(a) Valori calcolati mediante ρ.

(b) Valori calcolati mediante τ.

Figura 2.5: Valori di correlazione ottenuti eseguendo *BestSub* su una collezione di test di *TREC-8*, tratti da Guiver et al. [20, Figure 2 e 3]

che caratterizzano i sottoinsiemi migliori di topic (*Best*) individuati per la cardinalità corrente formano la curva blu, mentre i peggiori (*Worst*) formano quella rossa. Infine, i valori che caratterizzano i sottoinsiemi "medi", ossia quelli composti selezionando casualmente i topic (*Average*), formano la curva verde. Ciò che distingue le due figure è la formula usata per il calcolo di tali valori di correlazione. Nella figura 2.5a, infatti, viene utilizzata la ρ di Pearson, mentre nella 2.5b viene utilizzata la τ di Kendall.

I risultati ottenuti mostrano come i sottoinsiemi di topic *Best* riescano a prevedere in un modo decisamente migliore di quelli *Average* l'efficacia dei sistemi di IR sull'insieme originale. I sottoinsiemi *Worst*, inoltre, fanno molto peggio degli stessi sottoinsiemi *Average* e fra le due serie di valori vi è un ampio divario. Per fare un esempio prendendo in considerazione la ρ di *Pearson*, con sottoinsiemi *Best* da soli 8 topic si ottengono correlazioni più alte di 0.95; per ottenere lo stesso risultato con i sottoinsiemi *Average* risultano necessari almeno 23 topic, mentre con i sottoinsiemi *Worst* almeno 40. Tali risultati, inoltre, sono stabili al variare della metrica utilizzata. In Guiver et al. [20], infatti, vengono incluse metriche quali *R-Prec*, *P@10* e *logAP* (o *GMAP)*.

A questo punto è necessario precisare che la complessità computazionale del processo descritto risulta elevata e trovare delle soluzioni esatte diventa difficile anche per insiemi originali di topic caratterizzati da una dimensione n piuttosto piccola, poiché il numero di sottoinsiemi da analizzare aumenta esponenzialmente.

Per tale motivo, Guiver et al. [20] fanno uso di una ricerca euristica nella loro analisi, la quale viene descritta come segue:

> Un'euristica consiste nel cercare ricorsivamente: ossia, avendo identificato il miglior sottoinsieme di cardinalità c, nel cercare quello migliore di cardinalità $c + 1$ tra sottoinsiemi diversi per al massimo 3 topic (tale numero è stato scelto a priori poiché 4 topic sono ingestibili).

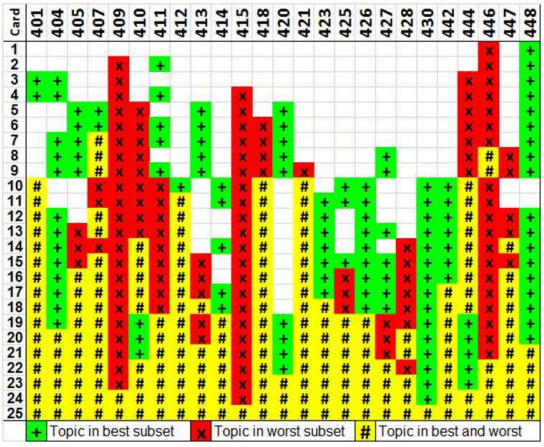
Gli effetti di tale euristica sui risultati vengono discussi nella sezione 2.3. Guiver et al. [20], inoltre, trattano due questioni strettamente collegate all'algoritmo euristico utilizzato, ossia:

1. quanta differenza si ottiene utilizzando i topic di un sottoinsieme *Best*/*Worst* di una data cardinalità c e quelli di un sottoinsieme della successiva cardinalità $c + 1$?

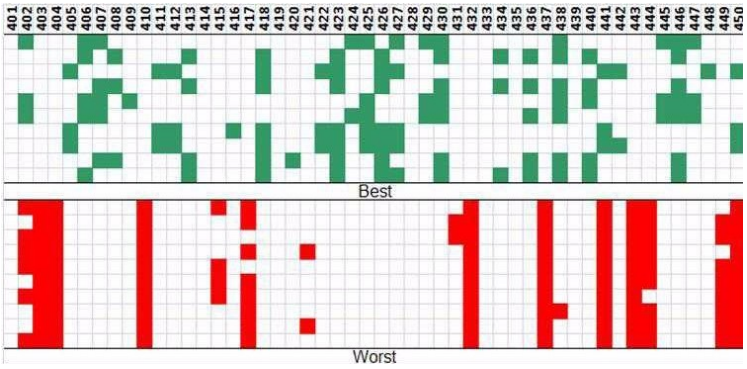
2. cosa si ottiene svolgendo un'analisi del vicinato, ossia selezionando anche il secondo sottoinsieme *Best*/*Worst*, il terzo, e quelli seguenti? Più precisamente, Guiver et al. [20] analizzano i primi dieci sottoinsiemi *Best*/*Worst* per ciascuna cardinalità.

I risultati di tali analisi vengono riportati nelle figure 2.6. In particolare, nella 2.6a è visibile una pixel-map topic per cardinalità per i valori di correlazione calcolati mediante ρ di Pearson. In ogni cella di tale mappa il valore può essere pari a `+` se per la relativa cardinalità il topic viene incluso nel sottoinsieme *Best*, `x` se viene incluso nel sottoinsieme *Worst* o `#` se viene incluso in entrambi i sottoinsiemi. Tale pixel-map, inoltre, mostra che in generale singoli topic sembrano essere "buoni" o "cattivi" e tale affermazione risulta vera, in particolare, per quelli "cattivi". Infatti, non appena un topic entra in un sottoinsieme *Worst* per una data cardinalità, esso tende a rimanere in tale sottoinsieme anche per quella seguente, mentre per il sottoinsieme *Best* si verificano alcune variazioni in tale schema.

Nella figura 2.6b è visibile una pixel-map topic per "bontà" (ossia, vengono riportati i primi sottoinsiemi *Best*/*Worst*, i secondi, i terzi, ecc.) per i valori di correlazione calcolati mediante ρ di Pearson, con cardinalità $c = 12$. In particolare, nelle prime dieci righe della mappa vengono mostrati altrettanti sottoinsiemi *Best*, mentre nelle ultime dieci vengono mostrati altrettanti sottoinsiemi *Worst*. Ogni cella viene colorata dii verde se il relativo topic fa parte di un sottoinsieme *Best*, nelle prime dieci righe della mappa stessa, mentre viene colorata di rosso se il relativo



(a) Pixel-map per l'intero dataset.



(b) Pixel-map per i migliori 10 sottoinsiemi con $c = 12$.

Figura 2.6: Pixel-maps costruite per analizzare la stabilità dei sottoinsiemi *Best*/*Worst* individuati da *BestSub*, con valori di correlazione calcolati mediante ρ, tratte da Guiver et al. [20, Figure 5 e 6].

topic fa parte di un sottoinsieme *Worst*, nelle ultime dieci. Tale mappa conferma che mentre il fatto di essere un topic "cattivo" è valido singolarmente, un sottoinsieme "buono" non è costituito solamente da topic "buoni" a livello individuale, ma più in generale da topic che in qualche modo contribuiscono alla previsione dell'efficacia dei sistemi di IR.

Il secondo contributo è quello di Robertson [30], il quale estende il lavoro di Guiver et al. [20] in quattro modi:

- Usa due collezioni aggiuntive: *TREC87* e *Terrier* (utilizzato per ottenere un'ulteriore popolazione di sistemi di IR sfruttando un insieme di diverse

configurazioni di *Terrier*[2]).

- Usa una metrica diversa per misurare l'efficacia dei sistemi di IR, ossia *logitAP*.
- Studia un metodo per costruire sottoinsiemi "buoni" di topic. In particolare, costuisce una matrice come proposto in Mizzaro et al. [25] per rappresentare le interazioni tra sistemi di IR e topic considerando, in particolare, la correlazione esistente tra la "facilità" dei topic stessi e la loro abilità nella previsione dell'efficacia dei sistemi di IR. Su tale matrice, utilizzando l'algoritmo *HITS* di Kleinberg [22], calcola una misura del valore di *hubness* di ciascun topic ed analizza il fatto che tale misura possa essere utilizzata o meno per trovare un sottoinsieme più piccolo di topic "buoni".
- Svolge tre esperimenti per generalizzare i risultati originali di Guiver et al., i quali consistono in (Robertson [30, p. 138]):
  1. un'analisi *HITS*: confronta i vettori di *hubness* dei topic provenienti dalle tre collezioni, secondo una data metrica;
  2. un'analisi dei sottoinsiemi *Best*/*Worst*: usa i sottoinsiemi *Best*/*Worst* calcolati sui dati di *TREC* sulla collezione di *Terrier*;
  3. una strategia di selezione dei topic: usa l'analisi *HITS* dei dati di *TREC* per prevedere sottoinsiemi "buoni" di topic su *Terrier*.

Le conclusioni delle analisi indicate confermano che scegliere un sottoinsieme "buono" di topic non ha a che fare solamente con la scelta di topic "buoni" a livello individuale.

Berto et al. [4] generalizzano il lavoro di Guiver et al. [20] e Robertson [30] in tre modi:

- investigando relativamente a quanti sottoinsiemi "buoni" di topic esistono;
- estendendo l'analisi svolta da Robertson [30] per generalizzare i risultati ottenuti da esso considerando varie metriche per la valutazione dell'efficacia dei sistemi di IR;
- estendendo i risultati originali di Guiver et al. [20] riguardanti i dieci sottoinsiemi *Best* di topic includendo un esperimento per generalizzare tale aspetto. In particolare, si chiedono cosa si verifica considerando il rango dei migliori sottoinsiemi di topic ed un sottoinsieme diverso da quelli individuati.

[2] `http://terrier.org/`

I risultati ottenuti mostrano che:

- esistono molti sottoinsiemi "buoni" di topic, per cui vi è la speranza che alcuni di essi lo siano in generale;
- anche se un singolo sottoinsieme "buono" non può essere generalizzato per una nuova popolazione di sistemi di IR, alcuni di quelli successivi possono risultare adeguati sotto certe condizioni;
- la metrica scelta ha un impatto decisivo quando si ha a che fare con i sottoinsiemi *Best*.

Si noti, infine, che i risultati di Guiver et al. [20], Robertson [30] e Berto et al. [4] sono *a posteriori*. Ciò significa che le loro analisi sono state svolte quando l'intero processo di valutazione risultava già terminato. Per tale motivo, tali risultati non sono direttamente applicabili per ottenere una strategia pratica di selezione dei topic in grado di identificare i sottoinsiemi "buoni" di topic *a priori* o, quantomeno, durante il processo di valutazione della *relevance* dei documenti recuperati dai sistemi di IR.

2.3 Il Software BestSub

BestSub è il software che ha permesso a Guiver et al. [20] di ottenere i risultati descritti nella sezione precedente, ma non è esente da alcune limitazioni, discusse nel seguito.

Una di tali limitazioni consiste nel fatto che l'euristica utilizzata è piuttosto grezza. Nell'implementazione di *BestSub*, l'effetto collaterale di tale euristica è che i sottoinsiemi di topic di cardinalità c e $c+1$ differiscono per al massimo k elementi. Più k è vicino ad 1, più il processo di ricerca diventa un algoritmo *greedy*. Guiver et al. [20] impostano tale parametro con un valore massimo pari a 3, mentre Robertson [30] investiga i risultati degli esperimenti svolti con un valore per il parametro k vicino a 1. Per quanto riguarda la ricerca esaustiva dei sottoinsiemi di topic, Guiver et al. [20] affermano che:

> quando viene utilizzata la τ di Kendall, cercare esaustivamente richiede circa 7 giorni con $c = 11$ e circa 20 giorni con $c = 12$, anche sfruttando il calcolo efficiente (O $n \log n$) della τ di Kendall svolto utilizzando l'algoritmo di Knight [Boldi et al. 2005]. La ricerca esaustiva svolta calcolando la correlazione con la formula standard è molto più veloce per via dei

calcoli più semplici e vi sono spazi più ampi per l'ottimizzazione dell'algoritmo; comunque, anche in questo caso, la computazione diventa un problema reale oltre $c = 15$.

Risulta particolarmente preoccupante il fatto che l'euristica possa distorcere i risultati di stabilità. Nonostante i valori di correlazioni ottenuti, probabilmente, non vengano influenzati pesantemente dall'uso dell'euristica, alcuni effetti sulla stabilità stessa (Figure 2.6) non possono essere esclusi ed è piuttosto probabile che tali effetti si concretizzino.

Infine, l'efficienza di *BestSub* non è ideale. Il suo algoritmo di ricerca, anche se ottimizzato, si traduce in una ricerca estremamente lenta, anche per insiemi originali molto piccoli di topic. Con il parametro $k = 3$ su un PC di fascia alta (Mac Pro del 2013) per 50 topic l'algoritmo impiega circa 10.5 ore per terminare, e per 250 topic con $k = 2$ (ossia, una ricerca pressoché *greedy*), esso stesso impiega più di un mese per terminare la sua computazione.

Nel capitolo 3 viene presentata la tecnica meta-euristica degli *Algoritmi Genetici*, i cui strumenti permettono di superare le limitazioni descritte in questa sezione ottenendo una ricerca estremamente più rapida che sfrutta una nuova euristica in grado di evitare l'effetto collaterale che caratterizza quella di *BestSub*.

Capitolo 3

Algoritmi Genetici

Lo scopo di questo capitolo è presentare la tecnica meta-euristica degli *Algoritmi Genetici*, che può essere considerata come una specializzazione di quella degli *Algoritmi Evolutivi*. In particolare, nella sezione 3.1 viene presentata una descrizione in termini generali di tale tecnica, mentre nella sezione 3.2 viene descritta la rappresentazione scelta per la popolazione di individui analizzata da *NSGA-II*. Tale algoritmo genetico, descritto nella sezione 3.6, è quello che viene utilizzato nell'implementazione della nuova versione di *BestSub*. Successivamente, le sezioni 3.3, 3.4 e 3.5 vengono dedicate alla descrizione degli operatori che un qualsiasi algoritmo genetico sfrutta per manipolare gli individui che analizza. Infine, la sezione 3.7 descrive il modo in cui è possibile ridurre il costo della valutazione dei sistemi di IR sfruttando gli Algoritmi Genetici.

3.1 Descrizione

Una *meta-euristica* è una tecnica *euristica* progettata per risolvere una classe molto ampia di problemi combinando diversi algoritmi a loro volta *euristici*, con l'obiettivo di produrre una procedura finale più efficiente da sfruttare qualora quelle standard non risultino in grado di giungere ad una soluzione finale o, comunque, di giungervi in un tempo accettabile. A sua volta, un *algoritmo euristico* è progettato per risolvere un problema più efficientemente ottenendo una soluzione *approssimata* che corrisponda ad un *ottimo locale* il più possibile vicino a quello *globale* all'interno dello *spazio di ricerca* delle soluzioni stesse. In generale, dunque, si cerca di ottenere un risultato che sia un compromesso tra ottimalità, completezza, accuratezza e velocità d'esecuzione. L'aspetto interessante di tale definizione consiste nel fatto che

vi sono due livelli di astrazione, quello interno ai singoli algoritmi euristici e quello dato dalla combinazione di tali algoritmi e di procedure più generali di passaggio dei dati tra gli stessi e quant'altro. Le idee che stanno alla base di una tecnica metaeuristica dipendono per la maggior parte dal problema che è necessario risolvere, tuttavia è possibile categorizzarle in base ai principi ai quali esse si ispirano. Uno di tali principi consiste nell'adottare un approccio alla risoluzione dei problemi ispirato al processo biologico dell'*Evoluzione*. Gli algoritmi che si basano su tale approccio vengono definiti *Algoritmi Evolutivi*.

Gli *Algoritmi Genetici*, sulla base delle definizioni fornite da Serafini [33] ed Eiben et al. [11], possono essere considerati come una specializzazione degli *Algoritmi Evolutivi* poiché integrano le idee di *selezione naturale* ed *eredità genetica* a quelle sfruttate dai secondi. In particolare, il problema che ciascuna specie deve affrontare in natura è quello della sopravvivenza, il quale consiste nel riuscire a trovare un adattamento vantaggioso ad un ambiente mutevole e talvolta ostile. Ognuna di tali specie, inoltre, è costituita da una *popolazione* di *individui*, dove ogni individuo è caratterizzato da un *patrimonio genetico* costituito da *cromosomi*, a loro volta formati da *geni*. Per poter evolvere e sopravvivere, esse devono subire delle *trasformazioni* che alterino tale *patrimonio genetico* contenuto nei *cromosomi* dei singoli individui.

L'idea sfruttata dagli *Algoritmi Genetici* consiste nell'allineare il loro procedimento all'analogia descritta nel paragrafo precedente generando una popolazione di individui, dove ognuno di essi corrisponde ad una delle possibili soluzioni del problema che l'algoritmo stesso deve risolvere. In figura 3.1 è possibile osservare uno schema riassuntivo delle principali fasi del flusso d'esecuzione di un tale algoritmo. Tali fasi, inoltre, vengono rappresentate ad un livello più implementativo dallo pseudocodice 3.1. La qualità di ogni soluzione viene valutata secondo il grado di *adattamento* all'ambiente in cui l'individuo cerca di sopravvivere mediante una cosiddetta *fitness function*, la quale può essere anche *multi-obiettivo*. All'interno della popolazione generata, verosimilmente, vi sono soluzioni di maggiore o minore qualità e per preservare la specie, dunque, viene svolta un'operazione di *Selezione* degli individui migliori, che consiste nell'accoppiare per la riproduzione solamente quelli la cui fitness function ha un valore superiore ad una determinata soglia. Successivamente, l'algoritmo accoppia concretamente tali individui secondo un qualche criterio in modo che diano alla luce nuove soluzioni, con la speranza che siano di maggior qualità. In altre parole, si spera che i figli si siano adattati meglio dei genitori all'ambiente in cui vivono. Tale operazione viene definita come *Crossover*.

Quanto descritto fino a questo punto si basa sull'assunzione che se si accoppiano

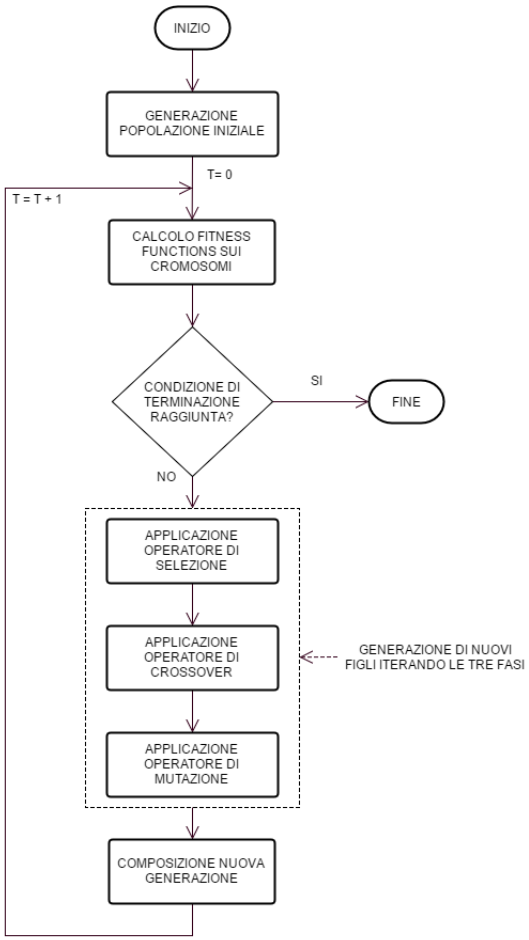


Figura 3.1: Flusso d'esecuzione generale di un algoritmo genetico.

soluzioni buone, si ottengono ancora soluzioni buone ma può verificarsi che i nuovi individui, di generazione in generazione, non migliorino più il loro patrimonio genetico, senza essere necessariamente peggiori dei genitori. Gli individui stessi, dunque, possono subire la *Mutazione* di uno dei loro geni, come risposta all'assenza di miglioramenti nell'adattamento all'ambiente, con la speranza che i loro figli riescano nuovamente a riprendere il processo di evoluzione della specie. Da una certa popolazione, dunque, se ne ricava un'altra, formata dagli individui appartenenti a quella originale e dai loro figli, e ciò viene ottenuto iterando le tre fasi precedenti. Tale composizione viene svolta secondo un qualche criterio che vada oltre ad una semplice operazione di unione poiché, solitamente, si sceglie di mantenere costante la dimensione complessiva della popolazione stessa riducendo il numero individui che

Algoritmo 3.1 Pseudocodice generale per un algoritmo genetico.

```
1:  P(O) ← GenerateInitialSolutions()
2:  t ← 0
3:  Evaluate(P(O))
4:  while not StoppingCriterion() do
5:      P1(t) ← Crossover(P(t))
6:      P2(t) ← Mutation(P1(t))
7:      P3(t) ← Selection(P2(t))
8:      Evaluate(P3(t))
9:      P(t+1) ← Update(P(t), P3(t))
10:     t ← t+1
11: end while
```

vengono effettivamente uniti.

La generazione della prima popolazione, il calcolo della fitness function, le operazioni di selezione, crossover e mutazione e la composizione della nuova generazione sono le fasi fondamentali nell'esecuzione di un algoritmo genetico i quali, in particolare, continuano a generare nuove popolazioni iterando tali fasi fino al raggiungimento di una condizione di terminazione predefinita. Tale condizione può consistere, ad esempio, in un limite al numero di generazioni successive o alla dimensione della popolazione complessiva. Un ulteriore motivo che porta a considerare gli algoritmi genetici come una specializzazione di quelli evolutivi consiste nel fatto che i primi svolgono tutte le operazioni dei secondi con l'aggiunta di quella di crossover. Gli algoritmi evolutivi non svolgono tale operazione poiché basano la loro attività, principalmente, sulla mutazione delle soluzioni generate di iterazione in iterazione.

3.2 Popolazione

Un aspetto particolarmente importante degli algoritmi genetici è scelta della *rappresentazione* da utilizzare per le soluzioni ammissibili, poiché da tale scelta dipende la fase di analisi e manipolazione delle soluzioni stesse. Una delle più utilizzate in letteratura consiste nel rappresentare i cromosomi degli individui come vettori di numeri interi e binari, dove ciascun numero rappresenta un gene del cromosoma. Tale rappresentazione ha il vantaggio di risultare piuttosto semplice da manipolare. La dimensione della popolazione costituita da tali individui, come già descritto cresce durante la fase di generazione iniziale e durante l'esecuzione dell'operazione di

crossover. Per quanto riguarda la fase iniziale, la dimensione minima richiesta per la popolazione è pari al numero di topic presenti nel dataset, per assicurare l'esistenza di almeno un sottoinsieme per ciascuna delle cardinalità da analizzare. Una volta raggiunto tale limite, i nuovi individui vengono generati casualmente ed in tal modo possono essere caratterizzati da una qualsiasi delle cardinalità ammissibili, al fine di ottenere più eterogeneità all'interno della popolazione stessa.

Vi è un ulteriore aspetto da prendere in considerazione, che riguarda il risultato restituito da un algoritmo genetico al termine di un'esecuzione. Come si è visto, tali algoritmi regolano il loro flusso d'esecuzione secondo obiettivi da raggiungere e ciò significa che nella maggior parte dei casi vi è uno spettro di soluzioni in grado di soddisfare tali obiettivi dove la scelta di quella finale risulta indifferente. Per tale motivo, un algoritmo genetico restituisce, solitamente, un *fronte di Pareto* di soluzioni ottime ed è ciò che avviene anche in questa tesi. Nelle sezioni 3.7 e 6.3 viene descritto, rispettivamente, come ciò si verifica e come tali soluzioni ottime vengono successivamente manipolate.

3.3 Operatore di Selezione

Solitamente, la strategia di selezione è ciò che distingue una tecnica metaeuristica dall'altra e, come si è visto nella sezione 3.1, il compito del relativo operatore consiste nello scegliere per l'attività riproduttiva solamente individui caratterizzati da un grado di adattamento all'ambiente superiore ad una certa soglia, dove tale grado viene determinato da una procedura che coinvolge il calcolo della fitness function. L'applicazione della strategia, in particolare, può essere divisa in due fasi distinte. La prima consiste nella definizione di una procedura per generare un insieme di N individui *buoni* tra i quali ne viene individuato uno che è *migliore* degli altri, secondo un qualche criterio. La seconda, invece, consiste analizzare tutte le coppie di individui appartenenti alla popolazione dove, per ciascuno dei membri della coppia, viene generato un nuovo insieme di individui *buoni* ed individuato quello *migliore* secondo la procedura definita nella fase precedente. Una volta fatto ciò, i due individui così ottenuti vengono utilizzati per l'attività riproduttiva. In altre parole, diventano l'input dell'attività di *Crossover*.

In questa tesi, la generazione dell'insieme di individui *buoni* tra i quali individuare quello *migliore* viene svolta con una procedura chiamata *Tournament Selection*. Tale procedura è visibile nello pseudocodice 3.2 e necessita di un parametro chiamato `k`, il quale corrisponde alla dimensione di tale insieme. Tale parametro, in altre pa-

Algoritmo 3.2 Pseudocodice per la procedura di *Binary Tournament Selection.*

```
1: best ← null
2: size ← SIZE(population)
3: for i=1 to k do
4:     individual ← population[RANDOM(1,size)]
5:     if best == null or CRITERION(individual) > CRITERION(best) then
6:         best ← individual
7:     end if
8: end for
9: return best
```

role, è il numero di *candidati* tra i quali scegliere l'individuo *migliore.* Si supponga, dunque, di porre `k=1`. Osservando lo pseudocodice 3.2, si può capire come tale l'individuo *migliore*, alla fine, sia semplicemente il primo ad essere individuato mediante una selezione casuale. Si supponga, a questo punto, di porre `k=n`, con `n` pari a dieci volte la dimensione della popolazione. In tal modo, si ha una probabilità piuttosto alta di selezionare ciascun membro di essa almeno una volta, perciò la procedura finisce per restituire sempre lo stesso individuo, il migliore in assoluto. Nessuna delle due situazioni, naturalmente, è desiderabile. Risulta necessario, dunque, un valore che permetta alla procedura di restituire individui *buoni* senza rischiare di prendere gli stessi più e più volte. In altre parole, un aumento della dimensione del torneo porta ad una perdita di *diversità* negli individui selezionati perché solo un ristretto sottoinsieme di essi contribuisce a tale diversità e l'operazione di selezione finisce per diventare un algoritmo *greedy.* In particolare, in Blickle et al. [5, pp. 368-372] si dimostra il legame esistente fra la dimensione del torneo e la percentuale di diversità persa ed il fatto che con un torneo che si svolge tra cinque candidati tale perdita è pari al 50%. Inoltre, si fa vedere come una dimensione pari a due o tre permetta di ottenere individui sufficientemente diversi ed una buona distribuzione dei valori della fitness function. Per tali motivi, risulta opportuno utilizzare uno di tali valori come dimensione del torneo. In questa tesi, il valore scelto per tale parametro è pari a due e la procedura risultante assume il nome di *Binary Tournament Selection.*

Rimane da definire, a questo punto, il criterio secondo il quale viene determinato l'individuo *migliore* tra quelli selezionati all'interno della procedura di *Binary Tournament Selection* il quale, nello pseudocodice 3.2, corrisponde alla chiamata alla funzione CRITERION(). Quello utilizzato, in particolare, è composto da due attributi distinti, i quali consistono nella definizione di una *rango di non dominazione* ed al

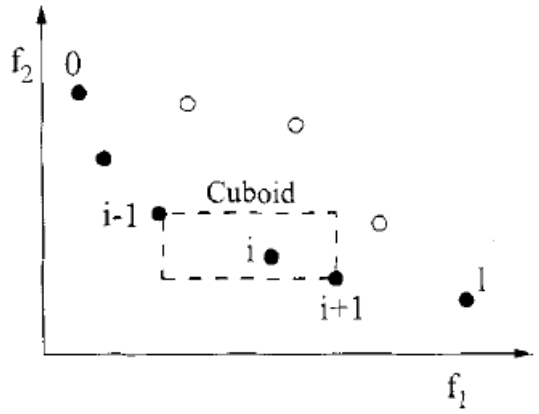


Figura 3.2: Rappresentazione del calcolo della *Crowding Distance*, tratta da Deb et al. [9, p. 186].

calcolo della *Crowding Distance* tra gli individui analizzati.

Per quanto riguarda il primo attributo, si può affermare che una soluzione *domina* una seconda soluzione secondo la definizione 3.1 (Sezione 3.7) e l'applicazione di tale definizione su un insieme di soluzioni genera un *fronte di Pareto* di soluzioni *non dominate*. Iterando tale applicazione, si possono formare più fronti, dove il secondo viene costruito applicando tale definizione alle soluzioni *dominate* da quelle appartenenti al primo, e così via. In tal modo, è possibile definire per ogni membro dell'insieme di individui *buoni* tale *rango di non dominazione*, il quale corrisponde all'indice del *fronte di Pareto* a cui appartiene.

Il secondo attributo corrisponde ad una stima della *densità* delle soluzioni che circondano una particolare soluzione. Per calcolare tale stima, si immagini di costruire un grafico nello spazio dato dalle componenti della fitness function. Ogni punto sul grafico stesso viene individuato dal calcolo di tali componenti su una singola soluzione. Per ognuno di essi viene calcolata lungo ogni asse la distanza media tra quello corrente ed i rimanenti. Tali quantità vengono usate per stimare il perimetro del *cuboide* formato utilizzando i punti più vicini a quello analizzato come vertici. La *Crowding Distance*, dunque, è la lunghezza media dei lati del cuboide definito dai vicini appartenenti allo stesso fronte di soluzioni non dominate. Nella figura 3.2 è possibile vedere una rappresentazione di tale calcolo nella quale la fitness function ha due componenti ed i punti pieni rappresentano soluzioni che appartengono allo stesso fronte non dominato.

Ricapitolando, ogni membro degli insiemi di individui *buoni* generati dalla pro-

cedura di *Binary Tournament Selection* eseguita su entrambi i membri della coppia correntemente analizzata possiede due attributi, il *rango di non dominazione* e la *Crowding Distance*. Dato un individuo x qualsiasi, tali attributi vengono chiamati, rispettivamente, x_{rank} e x_{dist} e sfruttandoli è possibile definire un operatore $\prec_c$ il quale stabilisce un *ordinamento parziale* nella forma $i_{rank} < j_{rank} \vee (i_{rank} = j_{rank} \wedge i_{dist} > j_{dist})$, dove i e j sono due individui diversi. Tale operatore, in particolare, tende a favorire soluzioni con un rango di dominazione minore quando si trovano su fronti diversi[1], ma quando appartengono entrambe allo stesso fronte viene selezionata quella appartenente ad una regione meno *densa* dello spazio di ricerca, ovvero quella per la quale la *Crowding Distance* è maggiore. Lo sfruttamento di tale operatore, dunque, rende possibile ordinare gli individui stessi e trovare quello *migliore*, guidando il processo di selezione verso un fronte di Pareto uniforme.

Il criterio di selezione analizzato è stato introdotto da Deb et al. [9], mentre Fortin et al. [17] propongono delle modifiche per risolvere alcuni problemi che incidono sull'attività dell'operatore $\prec_c$, i quali non vengono descritti in questa tesi.

3.4 Operatore di Crossover

Il compito dell'operatore di *Crossover*, come già precisato, consiste nell'accoppiare due o più individui di una popolazione per generare nuove soluzioni, con la speranza che siano migliori di quelle di partenza. Per svolgere tale operazione è necessario definire il modo in cui l'operatore sceglie i genitori da utilizzare per creare i nuovi individui e come quest'ultimi vanno effettivamente costruiti a partire dai genitori stessi. Nella sezione 3.3 si è visto il modo in cui i genitori vengono scelti sfruttando la procedura di *Binary Tournament Selection*. Per quanto riguarda la costruzione della prole, una volta selezionati i genitori l'operatore genera esattamente due figli. Il primo viene ottenuto ponendo i geni dei genitori stessi in AND, mentre il secondo ponendo i geni in OR. In tal modo essi contengono, rispettivamente, l'intersezione e l'unione dei topic degli individui originali. Nella figura 3.3, inoltre, è possibile vedere un esempio del procedimento svolto dall'operatore.

L'operatore di *Crossover* agisce nel flusso visibile nella figura 3.1 con un certo valore di probabilità impostato come parametro, al fine di poterlo modificare in qualsiasi momento, e l'impostazione di tale parametro dev'essere effettuata anche per l'operatore di *Mutazione*, come si vede nella sezione 3.5. Vi sono, per tale motivo, due elementi da definire. Il primo consiste nella scelta del procedimento secondo il

[1] In altre parole, preferisce le soluzioni *non dominate* individuate "prima".

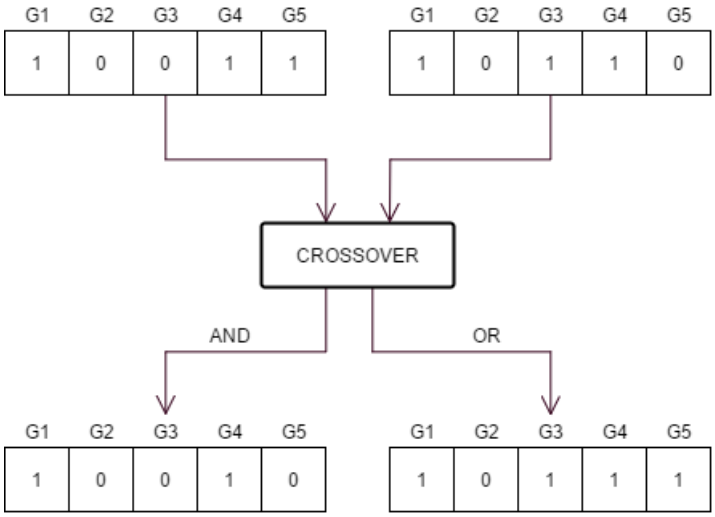


Figura 3.3: Esempio di applicazione dell'operatore di *Crossover*.

quale essi manipolano effettivamente le soluzioni, mentre il secondo consiste nella scelta dei valori di probabilità con cui regolare l'attività degli operatori stessi. Per quanto riguarda la seconda questione si possono individuare due approcci principali, secondo la classificazione effettuata in Eiben et al. [12]. Il primo di tali approcci è quello del *Parameter Tuning*, dove i valori dei parametri vengono fissati *prima* dell'esecuzione di un dato algoritmo evolutivo e non subiscono alcun cambiamento durante tale esecuzione. Il secondo è quello del *Parameter Control*, dove i valori di tali parametri vengono impostati *durante* l'esecuzione di un dato algoritmo evolutivo. In altre parole, essi ricevono un valore nella fase di inizializzazione dell'algoritmo stesso il quale subisce delle modifiche mentre l'esecuzione procede. Oltre a tale forma di classificazione per i due approcci all'impostazione dei parametri, in Eiben et al. [12] ed Eiben et al. [13] viene introdotta una tassonomia, viene presentata una panoramica sui possibili modi per implementare tali approcci e vengono discusse le relative questioni metodologiche.

Alla luce di quanto descritto nel paragrafo precedente, il modo utilizzato in questa tesi per assegnare un valore di probabilità all'operatore di *Crossover* al fine di regolarne l'attività appartiene al *Parameter Tuning* e consiste, semplicemente, nel fissare tale valore a priori seguendo un principio intuitivo per il quale *è bene generare molti figli, ma essi devono subire poche mutazioni genetiche.* In altre parole, l'operatore deve avere un valore di probabilità piuttosto alto, indicativamente tra 0.7 e 0.9. L'accoppiamento dei genitori per costruire i figli, infatti, è la fase che

permette alla dimensione della popolazione di crescere, aumentando così l'ampiezza dello spazio di ricerca delle soluzioni ammissibili. Tale aumento, inoltre, permettere di avvicinarsi alla condizione di terminazione dell'algoritmo la quale, in questo caso, corrisponde ad una limitazione superiore sulla dimensione della popolazione. Inoltre, un valore di probabilità alto tende a "trattenere" gli individui su un minimo locale o su un massimo, a seconda dei casi. I valori da assegnare ai parametri, in ogni caso, dipendono strettamente dal problema trattato. Quello utilizzato in questa tesi, dunque, è un approccio sostanzialmente empirico che fa parte di una categoria più ampia di metodi ed ha portato fin da subito all'ottenimento di ottimi risultati. Per tale motivo, non è risultato necessario uno studio che porti ad ottenere dei valori di probabilità migliori attraverso l'utilizzo di modelli più elaborati appartenenti alle due categorie precedentemente citate per gli operatori di *Crossover* e *Mutazione*. Ad ogni modo, tale studio rimane un interessante sviluppo futuro.

3.5 Operatore di Mutazione

Durante le iterazioni dell'algoritmo genetico può verificarsi che gli individui presenti in una generazione successiva non apportino miglioramenti a quella formata dai genitori, come descritto nella sezione 3.1. In altre parole, la specie non riesce più ad evolvere per adattarsi all'ambiente. L'analogia con il processo evolutivo descrive la necessità di introdurre una *mutazione genetica* in parte degli individui della specie stessa al fine di far ripartire tale processo, alterando alcuni dei geni che formano i cromosomi di tali individui. In altre parole, è necessario applicare modifiche casuali a parte delle soluzioni. Anche per l'operatore di *Mutazione*, dunque, è necessario definire come scegliere la soluzione da mutare e come effettuare concretamente tale operazione. In particolare, la soluzione da mutare viene scelta casualmente, poiché qualora l'operatore modificasse una soluzione di proposito finirebbe per pregiudicare il processo di valutazione delle stesse, rendendolo troppo "di parte". Per quanto riguarda l'applicazione della mutazione, essa viene effettuata eseguendo una scansione del vettore binario che caratterizza la soluzione scelta da sinistra verso destra decidendo, per ciascuno dei bit facenti parte del vettore, se invertirne o meno il valore, secondo un dato valore di probabilità. Si tratta di una strategia classica, descritta in Eiben et al. [11], e nella figura 3.4 è possibile osservarne un esempio.

Anche l'operatore di *Mutazione* agisce secondo un valore di probabilità ricevuto esternamente, indipendente da quello generato durante lo svolgimento dell'operazione. In questa tesi, tale valore viene assegnato secondo il principio intuitivo già

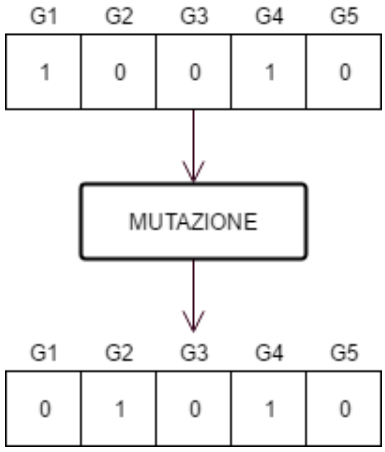


Figura 3.4: Esempio di applicazione dell'operatore di *Mutazione*.

descritto nella sezione 3.4, per il quale *è bene generare molti figli, ma essi devono subire poche mutazioni genetiche*. In altre parole, l'operatore deve agire secondo un valore di probabilità piuttosto basso, indicativamente compreso fra 0.1 e 0.3. Tale operatore, infatti, permette di ampliare lo spettro dell'esplorazione dello spazio di ricerca uscendo dal vicolo cieco dato dal minimo o massimo locale verso il quale l'algoritmo genetico viene spinto dall'operatore di *Crossover*, ma se viene svolta troppo frequentemente l'euristica complessiva degrada in una semplice *ricerca casuale*, vanificando gli sforzi fatti.

3.6 NSGAII

L'algoritmo appartenente alla tecnica meta-euristica degli algoritmi genetici utilizzato in questa tesi è la seconda versione del *Non-dominated Sorting Genetic Algorithm* (*NSGA-II*) e la descrizione che viene svolta nel seguito si basa sull'articolo originale di Deb et al. [9]. Si tratta di un evoluzione dell'algoritmo *NSGA* originariamente pubblicato da Deb et al. [10], costruita per risolvere tre aspetti particolarmente criticati della versione originale. In particolare, il primo di essi ha a che fare con l'elevata complessità computazionale della procedura utilizzata di ordinamento della popolazione in diversi fronti di soluzioni non dominate, mentre il secondo consiste nella mancata applicazione del concetto di *elitarismo* il quale, se sfruttato a dovere, può velocizzare significativamente le performance dell'algoritmo ed impedire la perdita di soluzioni buone, una volta individuate. Infine, il terzo aspetto consiste nella

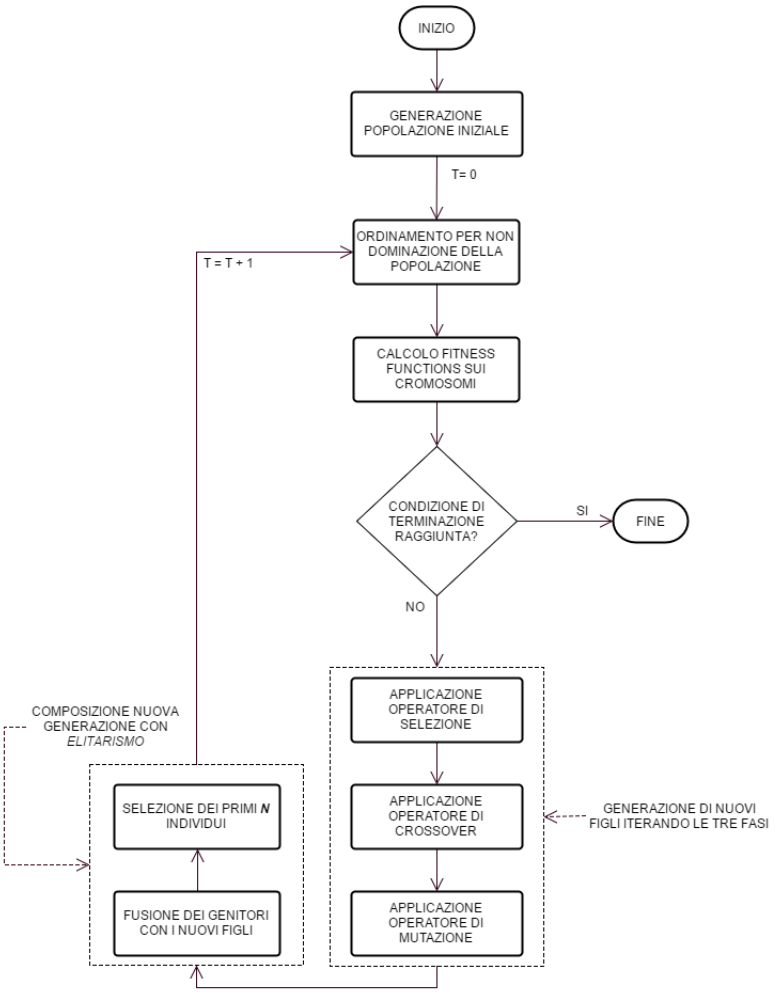


Figura 3.5: Flusso d'esecuzione di *NSGA-II*.

necessità di dover specificare manualmente un parametro per assicurare la diversità nella popolazione generata.

NSGA-II espande la struttura di base del flusso d'esecuzione di un algoritmo genetico visibile nella figura 3.1 ottenendo quella nella figura 3.5 ed utilizza per ciascun individuo la rappresentazione descritta nella sezione 3.2, mentre gli operatori agiscono secondo quanto descritto nelle sezioni 3.3, 3.4 e 3.5. La caratteristica fondamentale di tale algoritmo è il modo in cui genera *le popolazioni successive a quella di partenza* sfruttando una procedura veloce di ordinamento per *non dominazione*, il calcolo della *Crowding Distance*, l'operatore $\prec_c$ e lo sfruttamento del concetto di *elitarismo* precedentemente citato. Tale concetto consiste nel copiare una piccola parte

degli individui con i migliori valori di fitness function nella generazione successiva senza subire alcuna mutazione. Ciò può avere un grosso impatto sulle performance dell'algoritmo perché così esso non spreca altro tempo scoprendo nuovamente soluzioni già individuate in precedenza. Le soluzioni copiate senza mutazioni rimangono comunque valide come genitori da sfruttare per generare i nuovi individui.

La descrizione del flusso d'esecuzione di *NSGA-II* è tratta da Deb et al. [9, pp. 185-186]. Inizialmente, l'algoritmo genera una popolazione P_0 casuale, senza alcun vincolo interno su come svolgere tale operazione. Tale popolazione viene ordinata per *non dominazione* e per ognuna delle soluzioni viene calcolato il *rango di non dominazione*, il quale corrisponde all'indice del fronte non dominato d'appartenenza, come descritto nella sezione 3.3. Successivamente, vengono applicati gli operatori di *Selezione*, *Crossover* e *Mutazione* per creare un insieme di figli Q_0 di dimensione pari a N. A questo punto termina la prima fase dell'algoritmo e viene avviata la procedura per costruire le generazioni successive. Tale punto di svolta nell'esecuzione dell'algoritmo avviene a causa dell'applicazione dell'*elitarismo*. Infatti, per applicare tale concetto è necessario confrontare la popolazione corrente con le soluzioni *non dominate* individuate durante la generazione precedente. Per tale motivo, è necessario differenziare la costruzione della prima generazione e da quella delle successive.

La procedura di generazione della t-esima popolazione avviene secondo lo pseudocodice 3.3. Inizialmente, la generazione di partenza P_t ed i figli ottenuti Q_t vengono uniti[2] per formare la popolazione R_t di dimensione $2N$. Successivamente, R_t viene ordinata per *non dominazione*. Poiché tutti i figli vengono uniti ai genitori, una parte di essi sopravvive senza subire mutazioni ed il criterio dell'*elitarismo* viene soddisfatto. A questo punto è necessario ridurre la dimensione della popolazione fino a N per mantenere tale parametro costante fino all'applicazione degli operatori. Inizialmente, vengono mantenute le soluzioni localizzate nel miglior fronte non dominato, ovvero F_1. Se la dimensione di F_1 è minore di N, l'intero fronte viene incluso nella nuova popolazione P_{t+1}. I membri rimanenti della popolazione stessa vengono scelti dai fronti seguenti, ordinati per *rango di non dominazione*. In altre parole, vengono prese le soluzioni appartenenti a F_2, seguite da quelle in F_3, ecc. In generale, F_l è l'ultimo fronte le cui soluzioni vengono incluse nella popolazione P_{t+1}. Poiché ciascun fronte viene preso nella sua interezza, il numero di soluzioni contenute nei fronti $F_k : k \in [1, ..., l]$ è $\geq N$. Per riuscire a mantenere costante la dimensione della popolazione di generazione in generazione, il fronte F_l viene ordinato median-

[2]Durante la prima iterazione della procedura, sono P_0 e Q_0.

Algoritmo 3.3 Pseudocodice per la procedura di generazione della t-esima popolazione durante l'esecuzione di *NSGA-II*.

1: $R_t \leftarrow P_t \cup Q_t$
2: $(F_1, F_2, ..., F_k) \leftarrow$ FastNotDominatedSort(R_t)
3: $P_{t+1} \leftarrow \emptyset$
4: $i = 1$
5: **repeat**
6: CrowdingDistanceAssignment(F_i)
7: $i = i + 1$
8: $P_{t+1} = P_{t+1} \cup F_i$
9: **until** $|P_{t+1}| + |F_i| \leq N$
10: Sort$(F_i, \prec_c)$
11: $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_t + 1|)]$
12: $Q_{t+1} =$ GenerateOffspring(P_{t+1})
13: $t = t + 1$

te l'ordinamento parziale definito dall'operatore $\prec_c$ in ordine decrescente e da tale fronte vengono selezionate ed aggiunte alla nuova popolazione le prime m soluzioni, finché non vale che $|P_{t+1}| = N$. A questo punto, alla nuova popolazione P_{t+1} di dimensione N vengono applicati gli operatori di *Selezione*, *Crossover* e *Mutazione* per costruire Q_{t+1}, ossia la definitiva t-esima generazione di individui. Nella figura 3.6 è possibile vedere una rappresentazione grafica di quanto svolto dalla procedura descritta.

Un ulteriore considerazione che si può trarre dalla procedura di generazione della t-esima popolazione consiste nel fatto che le operazioni di ordinamento delle soluzioni in fronti e l'applicazione dell'operatore $\prec_c$ descritto nella sezione 3.3 non viene svolta solamente durante l'uso di quello di *Selezione*. Durante l'attività di selezione, infatti, l'operatore $\prec_c$ viene utilizzato per determinare gli individui *migliori* durante la procedura di *Binary Tournament Selection* da sfruttare come genitori dei nuovi figli, mentre in quella di generazione della t-esima popolazione viene utilizzato per ordinare e ridurre la popolazione secondo il *rango di non dominazione* e la *Crowding Distance*. Un ulteriore effetto positivo dato dall'utilizzo dell'operatore consiste nel fatto che confrontare le soluzioni *non dominate* mediante una stima della densità del loro vicinato assicura una maggiore diversità delle stesse ed evita l'uso dell'ulteriore parametro che risulta necessario alla versione precedente di *NSGA-II*.

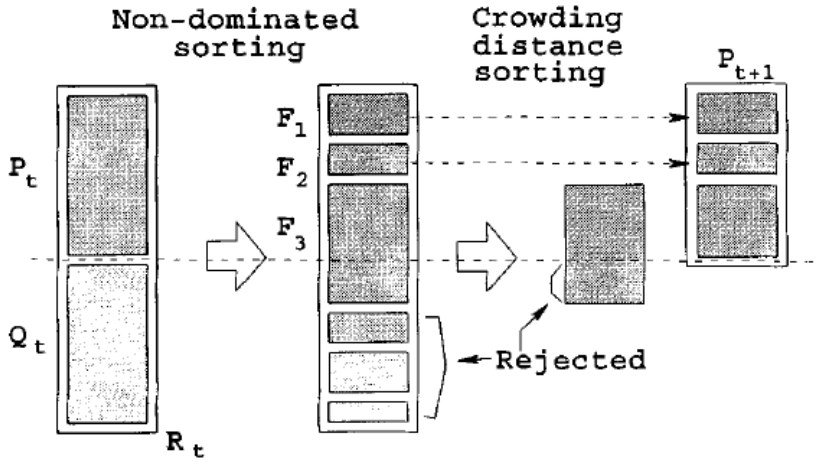


Figura 3.6: Rappresentazione della procedura di generazione della t-esima popolazione svolta durante l'esecuzione di *NSGA-II*, tratta da Deb et al. [9].

3.7 Riduzione del Costo della Valutazione Mediante Alg. Genetici

A questo punto è possibile descrivere il nuovo procedimento mediante il quale gli *Algoritmi Genetici* permettono di mettere in pratica l'approccio alla riduzione del costo della valutazione dei sistemi di IR studiato da Guiver et al. [20], Robertson [29] e Berto et al. [4] e descritto nella sezione 2.2. In particolare, la caratteristica fondamentale di tale nuovo procedimento consiste nel fatto che gli esperimenti *Best* e *Worst* si prestano particolarmente bene ad essere modellati come un *problema di ottimizzazione multi-obiettivo*. Problemi appartenenti a tale categoria possono essere facilmente forniti in input ad un dato algoritmo genetico il quale, successivamente, sfrutta l'euristica che lo caratterizza per risolvere il problema ricevuto ed ottenere un insieme di soluzioni ottime. Essendo algoritmi euristici, inoltre, permettono di ottenere quel compromesso necessario fra l'ottimalità delle soluzioni e l'efficienza nella loro individuazione. Nel caso di questa tesi, gli esperimenti *Best* e *Worst* rappresentano, più specificatamente, due istanze dello *stesso* problema di ottimizzazione multi-obiettivo, le quali vengono fornite in input ad *NSGA-II*. Ad ogni modo, prima di poter descrivere il modo in cui è possibile legare un'istanza di tale categoria di problemi ad un dato algoritmo genetico è necessario fornire una definizione precisa degli stessi.

Un problema di ottimizzazione multi-obiettivo presuppone, secondo la definizio-

ne fornita da Serafini [33, pp. 35-40], l'esistenza di un insieme di individui X, dove tali individui rappresentano decisioni che un dato decisore può prendere per risolvere il problema analizzato. Il decisore, dunque, dev'essere in grado di poter confrontare direttamente tutte le alternative possibili. Il fatto che tali individui rappresentino delle decisioni alternative porta alla definizione di quattro diverse relazioni che possono stabilirsi tra di esse. Si supponga, dunque, di scegliere $x \in X$ e $y \in X$ e si costituisca una coppia ordinata xRy dove R è una delle possibili relazioni. I casi che possono verificarsi sono quattro, ovvero: x *è preferita* a y, y *è preferita* a x, x e y sono *indifferenti* ed x e y sono *incomparabili*. Tutto questo porta alla definizione intuitiva 3.1, per la quale:

Definizione 3.1. Si dicono *non dominate* o *ottimi di Pareto* le decisioni x per le quali non esiste una decisione y preferita a x.

La situazione, in realtà, è più complessa in quanto ciascun decisore, solitamente, ha bisogno di valutare le alternative secondo diversi criteri per stabilire se una di esse viene preferita ad un'altra o risulta essere indifferente (o incomparabile). Si può immagine, dunque, di rappresentare ciascun criterio con una funzione obiettivo $f_i : X \to \mathbb{R}$ per la quale vale che che:

$$x \text{ è preferita a } y \iff f_i(x) < f_i(y) \tag{3.1}$$

$$x, y \text{ sono indifferenti} \iff f_i(x) = f_i(y) \tag{3.2}$$

Qualora ci fosse solo un criterio, le due condizioni precedenti risultano sufficienti poiché escludono l'incomparabilità e rendono possibile trovare una singola funzione obiettivo che le rispetti, e ciò costituisce una conseguenza desiderabile, altrimenti non sarebbe possibile alcune forma di decisione. Tuttavia, nel caso di criteri multipli, ammettere l'incomparabilità è positivo, poiché in tal modo il numero di decisioni non dominate aumenta. Tale principio, tuttavia, rende le condizioni 3.1 e 3.2 insufficienti.

Per risolvere un problema di decisione con criteri multipli rappresentati da un insieme F di funzioni obiettivo, dunque, è necessario le decisioni limitate ad uno solo di essi non danno luogo ad incomparabilità ma si presentano come un ordinamento totale, rispettando le condizioni 3.1 e 3.2, mentre per quelle eseguite prendendoli in considerazione tutti vale che $\forall f_i : f \in F, i \in \{1, \ldots, |F|\}$:

$$x \text{ è preferita a } y \Rightarrow f_i(x) < f_i(y) \tag{3.3}$$

$$x, y \text{ sono indifferenti} \Rightarrow f_i(x) = f_i(y) \tag{3.4}$$

In tal modo, è possibile trovare un ottimo locale secondo ciascun criterio, per poi giungere all'ottimo globale applicando le condizioni 3.3 e 3.4. Un *problema multi-obiettivo* può essere caratterizzato, dunque, sfruttando la definizione di Serafini [33, p. 38], per la quale «è un problema caratterizzato dalla presenza di un certo numero di funzioni obiettivo derivate dall'individuazione di criteri diversi sull'insieme delle soluzioni ammissibili».

Ricapitolando, si supponga nuovamente di prendere in considerazione un insieme X di soluzioni ammissibili, sul quale vengono definiti m diversi obiettivi nella forma di m funzioni obiettivo $f_i(x) : X \to \mathbb{R}$ da minimizzare. Componendo le quattro condizioni descritte e formalizzando le considerazioni qualitative del paragrafo precedente, si ottiene la seguente struttura:

$$
\begin{aligned}
&x \text{ è preferita a } y && \iff f_i(x) \leq f_i(y),\ i \in \{1, \ldots, m\},\ \text{e } f(x) \neq f(y) \\
&y \text{ è preferita a } x && \iff f_i(x) \geq f_i(y),\ i \in \{1, \ldots, m\},\ \text{e } f(x) \neq f(y) \\
&x, y \text{ sono indifferenti} && \iff f(x) = f(y) \\
&x, y \text{ sono incomparabili} && \iff \text{altrimenti}
\end{aligned}
$$

Applicando la definizione 3.1 a tale struttura si ottiene che una decisione x *domina* una decisione y se $\forall i \in \{1, \ldots, m\}, f_i(x) \leq f_i(y)$ ed $\exists k \in \{1, \ldots, m\} : f_k(x) < f_k(y)$, mentre una decisione x è *dominata* da una decisione y se $\exists i \in \{1, \ldots, m\} : f_i(x) > f_i(y)$. Una decisione x, perciò, è *non dominata* se non esiste alcuna decisione y che *domina* x. Assumendo che le funzioni obiettivo esprimano tutte le preferenze del decisore, quelle *non dominate* risultano essere le uniche decisioni da prendere.

A questo punto è possibile descrivere la caratteristica fondamentale che permette di legare un'istanza di un problema di ottimizzazione multi-obiettivo ad un dato algoritmo genetico. Tale caratteristica, in particolare, consiste nel fatto che *le funzioni obiettivo del problema diventano le componenti della fitness function multi-obiettivo dell'algoritmo.* Al fine di risolvere il problema che caratterizza questa tesi, dunque, è necessario individuare tali funzioni obiettivo da utilizzare come componenti della fitness function multi-obiettivo di *NSGA-II.*

Come descritto nella sezione 2.2, gli esperimenti *Best* e *Worst* mirano ad analizzare sottoinsiemi di topic per ciascuna cardinalità c, con la differenza che nell'ambito del primo esperimenti si intende trovare il sottoinsieme *migliore*, ossia quello con il più *alto* valore di correlazione con l'insieme originale di topic, mentre nell'ambito del secondo si intende trovare il sottoinsieme *peggiore*, ossia quello con il più *basso* valore di correlazione con l'insieme originale di topic. Da tale descrizione si può notare come i parametri fondamentali da prendere in considerazione siano due, os-

sia la cardinalità del sottoinsieme di topic correntemente analizzato ed il valore di correlazione esistente tra il suo valore di *MAP* (o di *NDCG*) e quello dell'insieme originale di topic. L'esperimento *Best*, dunque, mira ridurre il costo della valutazione dei sistemi di IR trovando un sottoinsieme di topic che *minimizzi* la cardinalità e *massimizzi* la correlazione, mentre l'esperimento *Worst* mira ad individare i sottoinsiemi di topic da *non* utilizzare per la riduzione del costo della valutazione dei sistemi di IR, ossia quelli che *massimizzano* la cardinalità e *minimizzano* la correlazione. Le funzioni obiettivo delle due istanze del problema, dunque, possono essere rappresentate nel seguente modo, dove x è un generico sottoinsieme di topic:

$$f_1(x) = \min \text{ card(x)} \quad f_1(x) = \max \text{ card(x)}$$
$$f_2(x) = \max \text{ corr(x)} \quad f_2(x) = \min \text{ corr(x)}$$

Tali funzioni obiettivo, come già precisato, diventano le componenti della fitness function multi-obiettivo utilizzata da *NSGA-II*. Tale algoritmo, dunque, utilizza le due componenti per svolgere entrambi gli esperimenti con un significato diverso per ciascuno di essi, sfruttando la nuova euristica che lo caratterizza. Una volta terminata l'esecuzione, esso restituisce un fronte di Pareto di soluzioni non dominate. L'utilizzo di una fitness function multi-obiettivo, inoltre, permette di migliorare l'efficienza della computazione poiché *NSGA-II* può, in tal modo, cercare soluzioni e valutare le due componenti della fitness function parallelamente.

Nel corso di tale descrizione non viene mai citato l'esperimento *Average*. Ciò è dovuto al fatto che tale esperimento può essere risolto direttamente poiché più facile e decisamente meno oneroso dal punto di vista computazionale degli esperimenti *Best* e *Worst*. A differenza di quest'ultimi, dunque, la fase di modellazione come problema di ottimizzazione multi-obiettivo da risolvere con *NSGA-II* non risulta necessaria.

Basandosi sulla descrizione del nuovo procedimento definito per mettere in pratica l'approccio alla riduzione del costo della valutazione dei sistemi di IR studiato da Guiver et al. [20], Robertson [29] e Berto et al. [4], inoltre, è possibile approfondire alcuni aspetti che caratterizzano la popolazione analizzata da *NSGA-II* e le manipolazioni svolte sugli individui di tale popolazione dagli operatori, poiché la natura degli individui stessi viene chiarita. Nel contesto di questa tesi, infatti, ognuno di tali individui rappresenta, come già indicato nei paragrafi precedenti, un sottoinsieme di topic di una data cardinalità e viene caratterizzato da un vettore binario di lunghezza pari al numero di topic contenuti nell'insieme originale, dove l'i-esima cella di esso indica se un determinato topic viene incluso nel sottoinsieme

stesso oppure no. Nella sezione 3.5, inoltre, viene descritto il modo in cui l'operatore di Mutazione svolge la propria attività. In questo caso, dunque, il significato è che quando viene mutato un individuo che rappresenta un sottoinsieme di topic, per ognuno di tali topic viene generato un valore compreso tra 0 ed 1 e se tale valore è minore di quello di probabilità fissato a priori quello correntemente analizzato il topic correntemente analizzato viene rimosso dal sottoinsieme o incluso in esso, se già presente o meno.

Nel capitolo 4 viene descritto *jMetal*, un framework orientato agli oggetti che fornisce l'implementazione di molti algoritmi genetici, tra le quali vi è anche quella di *NSGA-II*. Inoltre, si vede com'è possibile rappresentare concretamente problemi di ottimizzazione multi-obiettivo mediante il framework stesso ed il modo in cui lo si può integrare in un software esterno.

Capitolo 4

Il Framework jMetal

Lo scopo di questo capitolo è descrivere le caratteristiche di *jMetal*, il framework per l'ottimizzazione di problemi multi-obiettivo utilizzato nella costruzione di *NewBestSub*, il quale fornisce l'implementazione di diversi algoritmi appartenenti alla tecnica meta-euristica degli *Algoritmi Evolutivi*. Nella sezione 4.1, in particolare, vengono descritte le motivazioni ed i principi che guidano lo sviluppo del framework stesso e gli scopi per i quali può essere utilizzato, mentre nella sezione 4.2 ne viene discussa l'architettura interna ad un livello generale. Nelle relative sottosezioni, inoltre, vengono descritti dettagliatamente i vari elementi dei quali essa si compone ed il modo in cui un software esterno può essere integrato con jMetal. Infine, nella sezione 4.3 vengono riportate alcune precisazioni necessarie per completare la descrizione dell'architettura del framework stesso.

4.1 Motivazioni

jMetal [26, 27] è un framework orientato agli oggetti sviluppato in Java, del quale esistono diversi porting in altri linguaggi quali Python e C++. Il suo obiettivo principale è quello di permettere la risoluzione di problemi di ottimizzazione multi-obiettivo mediante l'utilizzo di tecniche meta-euristiche come quella degli *Algoritmi Evolutivi*. Tale tecnica, inoltre, non è l'unica disponibile all'interno del framework poiché sono presenti anche algoritmi di ottimizzazione con sciami di particelle, algoritmi ad evoluzione differenziale ed altro. Un ulteriore obiettivo legato allo sviluppo di tale framework consiste nella possibilità di rendere disponibili ai ricercatori che lavorano su problemi del genere sfruttando le tecniche precedentemente citate, all'interno di un software che sia il più possibile di facile utilizzo. Grazie a jMetal,

dunque, è possibile testare algoritmi allo stato dell'arte appartenenti alle varie tecniche meta-euristiche sui propri data set, contribuire sviluppando le proprie soluzioni oppure integrare jMetal all'interno di tool esterni per condurre analisi ed esperimenti di vario genere.

4.2 Architettura

Per quanto riguarda la struttura che permette a *jMetal* di rappresentare un problema di ottimizzazione multi-obiettivo, nella figura 4.1 è possibile osservare le quattro interfacce che compongono l'architettura di base del framework. Riportando quando spiegato da Nebro [26], tali interfacce catturano la scomposizione del problema in diverse parti le quali, interagendo vicendevolmente, consentono di giungere ad una o più soluzioni finali. Secondo lo schema architetturale, dunque, si ha che un *problema* sfrutta un *algoritmo* in grado di gestire un insieme di possibili *soluzioni* attraverso l'uso di diversi *operatori* che hanno il compito di manipolare un *numero variabile* di *soluzioni candidate* durante ciascuna iterazione.

Tale architettura è il risultato di un operazione di rinnovamento dell'intero framework descritta da Nebro et al. [28], che ha portato alla release della quinta versione del framework stesso. La quarta versione, infatti, soffriva di diversi problemi di progettazione i quali portavano ad avere un'architettura molto più complessa, con un'elevata difficoltà d'integrazione in eventuali software esterni. Uno dei migliora-

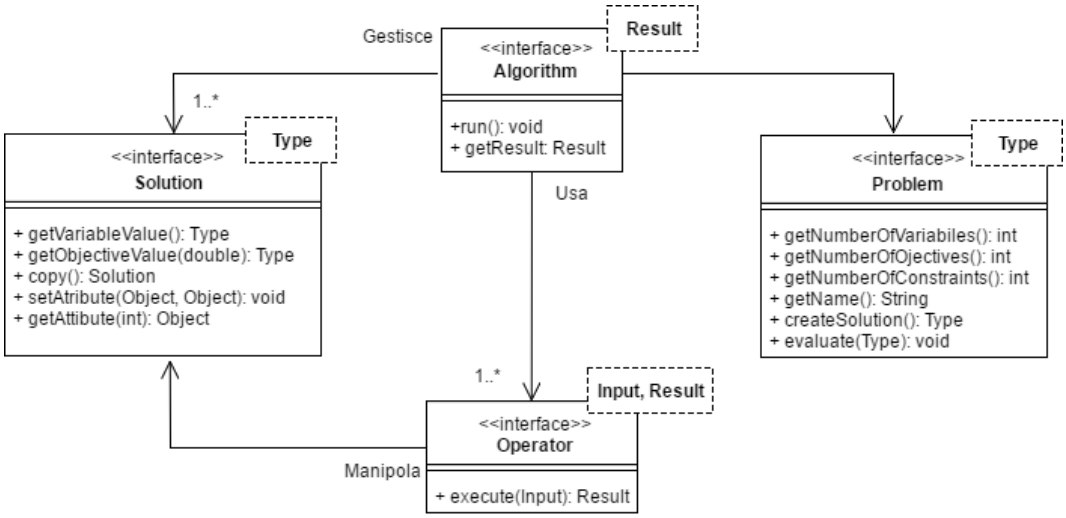


Figura 4.1: Interfacce che compongono l'architettura di base di *jMetal*.

menti della quinta versione, ad esempio, consiste nell'utilizzo di una sola entità per rappresentare le possibili soluzioni, mentre nella precedente ve ne potevano essere fino a cinque. L'uso dei tipi di dato generici, inoltre, ha permesso di eliminare un insieme di classi utilizzato per rappresentare tipologie diverse della stessa entità[1]. Il fatto di dover interagire con sole quattro interfacce porta ad una maggiore facilità di integrazione. Una delle prime attività svolte durante la fase di progettazione architetturale di *NewBestSub*, infatti, è stata quella di analizzare un'implementazione esemplificativa di un problema risolto con la quarta versione di jMetal e di implementare nuovamente tale esempio con la versione successiva del framework. La differenza fra le complessità progettuali delle due implementazioni è notevole, per cui è risultato conveniente procedere con la quinta versione dello stesso.

4.2.1 L'Interfaccia `Solution`

Una delle prime questioni con cui è necessario confrontarsi nella risoluzione di un problema di ottimizzazione con tecniche meta-euristiche è la scelta della modalità con cui rappresentare le soluzioni prodotte durante le varie iterazioni. Tale aspetto è fondamentale in quanto determina le operazioni che gli operatori possono svolgere su di esse ed influisce sul comportamento e sulle prestazioni della tecnica scelta. Nel framework l'interfaccia `Solution` ha tale compito ed, in realtà, è costituita da una gerarchia formata da quattro diverse entità, come si può vedere nella figura 4.2. Quella di base ha segnature per ottenere l'individuo (il framework lo definisce *variabile*) che costituisce la soluzione candidata ed i valori di ciascuna funzione obiettivo calcolata su di esso. Inoltre, è presente un'ulteriore segnatura la quale descrive la possibilità di ottenere una copia della soluzione stessa.

Le interfacce specializzate della gerarchia visibile nella figura 4.2 rappresentano soluzioni costituite da tre diverse tipologie di individui che possono essere manipolate in un problema di ottimizzazione, con segnature aggiuntive per ciascuna di esse. Tale gerarchia è stata progettata, dunque, per permettere di avere implementazioni diverse per lo stesso tipo di rappresentazione. `BinarySolution`, in particolare, rappresenta soluzioni dove gli individui assumono la forma di vettori binari e sono utili per rappresentare, ad esempio, delle configurazioni di elementi e tale interfaccia viene caratterizzata dalla segnatura di un metodo il quale consente di ottenere il numero di elementi all'interno di ciascun vettore. `DoubleSolution` e `IntegerSolution` vengono utilizzate quando gli individui di ciascuna soluzione sono singoli valori dou-

[1] Classi quali `SolutionType` e simili.

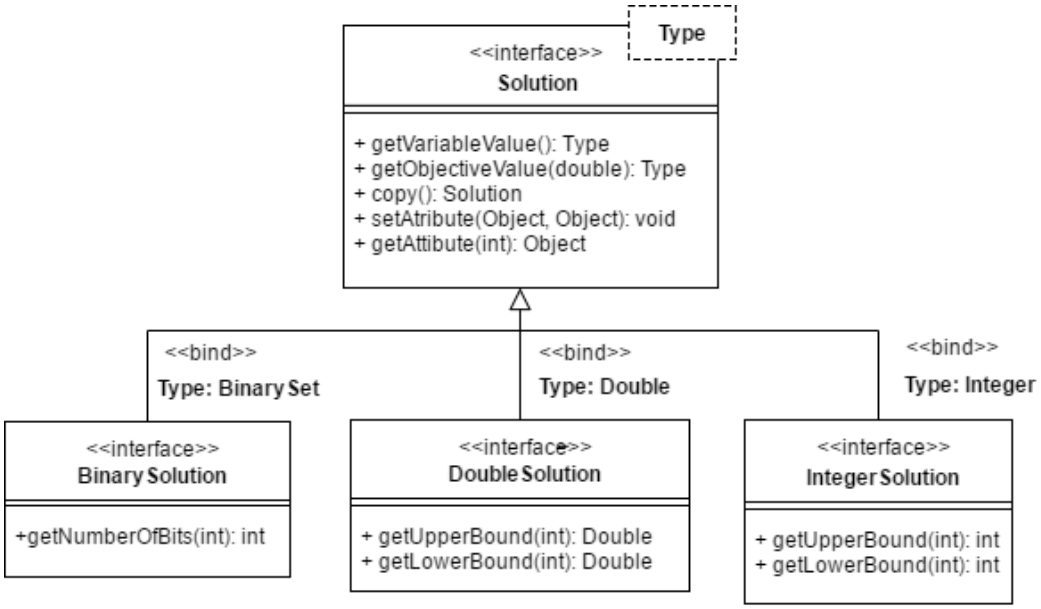


Figura 4.2: Gerarchia che specializza l'interfaccia `Solution` dell'architettura di base di *jMetal*.

ble (interi) o liste di essi, ed i due metodi aggiuntivi servono ad ottenere i limiti inferiori e superiori di tali liste. Tali interfacce risultano sufficienti per le soluzioni della maggior parte dei problemi. Qualora un particolare problema avesse la necessità di trattare soluzioni costituite, allo stesso tempo, da individui di tipo intero e double, è sufficiente implementare entrambe le interfacce ottenendone una definita come `DoubleIntegerSolution`, ad esempio.

4.2.2 L'Interfaccia `Problem`

Una volta definita la rappresentazione delle soluzioni è necessario definire il processo mediante il quale esse vanno inizializzate, per poi essere manipolate dagli operatori ed infine valutate, ossia il problema in quanto tale. L'elemento architetturale della figura 4.1 in cui vengono svolte le fasi di inizializzazione e valutazione delle soluzioni candidate è l'interfaccia `Problem`. Anche tale interfaccia viene specializzata da una gerarchia, come si può vedere nella figura 4.3. La prima cosa da fare per definire il problema consiste nell'implementare le segnature dei metodi getter in grado di fornire all'algoritmo utilizzato informazioni quali il numero di variabili di decisione, il numero di funzioni obiettivo, il nome del problema stesso ed, infine, il numero di vincoli che devono essere rispettati dall'algoritmo risolutivo.

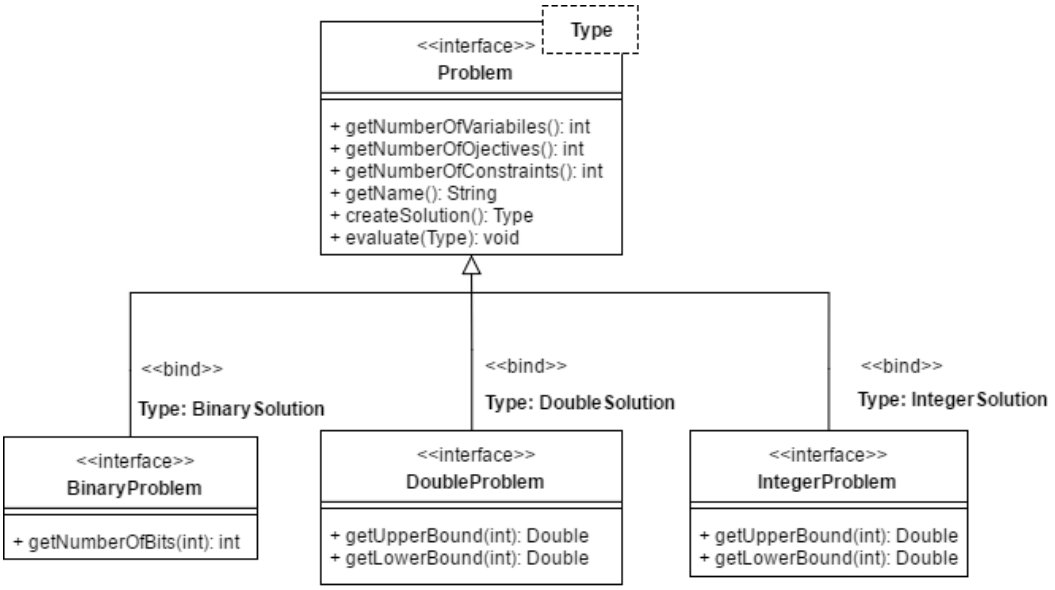


Figura 4.3: Gerarchia che specializza l'interfaccia `Problem` dell'architettura di base di *jMetal*.

Nel caso di questa tesi, vi è una sola variabile di decisione, poiché le soluzioni vengono caratterizzate da un singolo individuo il quale rappresenta un sottoinsieme di topic, mentre le funzioni obiettivo sono due, come descritto nella sezione 3.7. Per quanto riguarda eventuali vincoli per il problema, il framework li modella come limitazioni da imporre all'algoritmo risolutivo nella gestione delle soluzioni. Tuttavia, tale funzionalità non è stata utilizzata in questa tesi poiché è risultato necessario intervenire vincolando le fasi di generazione iniziale della popolazione e di valutazione delle soluzioni manipolate dall'algoritmo utilizzato ed, in ogni caso, ciò che viene svolto internamente si limita ad un tracciamento del numero di volte con cui occorrono violazioni di tali vincoli, senza fornire un meccanismo per la gestione di esse.

Le rimanenti segnature, infine, indicano la presenza di un metodo nell'ambito del quale vengono svolte le operazioni che possono essere necessarie prima della creazione di una nuova soluzione candidata e quella di un metodo in cui viene definito come dev'essere svolta la valutazione di una soluzione ottenuta dopo la fase di manipolazione svolta dagli operatori. Nell'implementazione concreta di tale metodo, dunque, vengono effettivamente impostati i valori delle funzioni obiettivo calcolate per la soluzione ricevuta. Come nel caso della creazione, potrebbero essere necessarie operazioni articolate prima di poter assegnare un valore alle funzioni stesse, per

cui il tutto viene incapsulato all'interno di tale metodo. Nel caso di questa tesi tale possibilità viene sfruttata ampiamente, in quanto sia in fase di creazione che di valutazione ci sono diversi aspetti da tenere in considerazione che vengono dettagliati nella sezione 6.3.

Le interfacce che specializzano la gerarchia in figura 4.3 rappresentano, similmente a quanto avviene per le soluzioni, problemi configurati per gestire una particolare tipologia di tali soluzioni. L'interfaccia `BinaryProblem` rappresenta, dunque, un problema le cui soluzioni vengono caratterizzate da individui nella forma di vettori binari e le interfacce `DoubleProblem` e `IntegerProblem` rappresentano, analogamente, problemi con soluzioni caratterizzati da individui di tipo intero o double. Tali interfacce specializzate possiedono segnature aggiuntive simili a quelle utilizzate per caratterizzare la tipologia di soluzione trattata. Il senso di esse è che i problemi devono poter comunicare le informazioni fornite dalla singola soluzione a livello globale. L'implementazione dei metodi aggiuntivi di `DoubleProblem` e `IntegerProblem`, ad esempio, consiste nel restituire un elemento da una lista di valori, dove ognuno di essi rappresenta la miglior limitazione inferiore o superiore per una delle soluzioni analizzate. La lunghezza di tali liste, dunque, è pari al numero di soluzioni utilizzate dal problema stesso.

4.2.3 L'Interfaccia `Operator`

Una volta definita la rappresentazione delle soluzione ed il modo in cui esse vanno create e valutate, è necessario definire quali operatori possono manipolarle e come ciò dev'essere svolto. L'elemento architetturale tra quelli presenti nella figura 4.1 che ha tale compito è l'interfaccia `Operator`. Tale interfaccia è molto semplice, ed indica la presenza di un metodo che esegue le operazioni previste dall'operatore su una *sorgente* che viene trasformata in un *risultato* e resa disponibile alla fase di valutazione del problema.

Anche tale interfaccia viene specializzata in una gerarchia, visibile nella figura 4.4. Poiché il framework incorpora al suo interno molti algoritmi genetici ed in questa tesi viene utilizzato uno di tali algoritmi, nel diagramma vengono illustrate le interfacce specializzate che rappresentano, in particolare, gli operatori sfruttati da tali algoritmi, ma esse non sono le uniche. Ad ogni modo, tali interfacce dettagliano maggiormente la natura della sorgente e del risultato manipolato dall'operatore e solo in alcuni casi aggiungono ulteriori segnature oltre a quella d'esecuzione. L'interfaccia `CrossoverOperator`, dunque, ha come sorgente e risultato una lista di `Solution`, poiché tale operatore ha il compito di creare delle soluzioni figlie par-

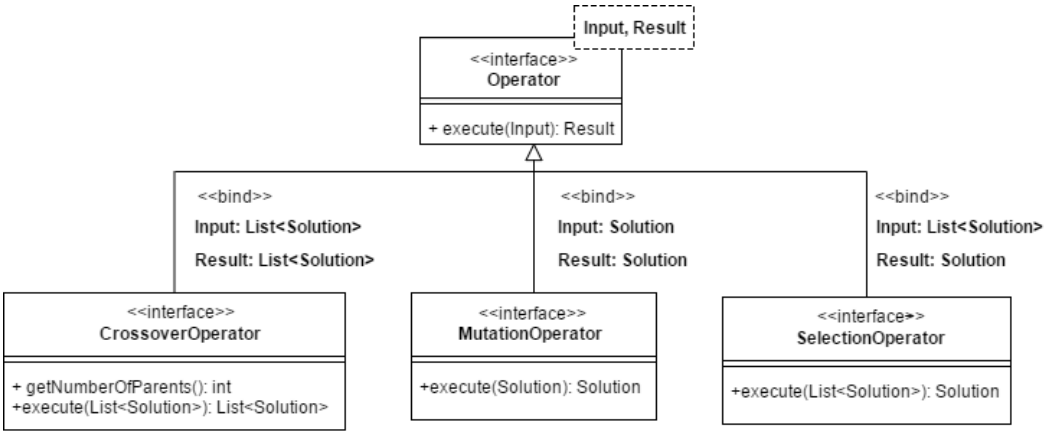


Figura 4.4: Gerarchia che specializza l'interfaccia `Operator` dell'architettura di base di *jMetal*.

tendo dai genitori forniti in input ed ha una segnatura aggiuntiva per definire il numero di genitori da prendere in considerazione. L'interfaccia `MutationOperator` riceve e restituisce una singola soluzione, poiché quest'ultima viene mutata secondo il metodo stabilito dall'operatore. Infine, `SelectionOperator` ottiene una lista di soluzioni, dalla quale sceglie quella da rendere disponibile alla fase di valutazione del problema. Specializzando l'interfaccia `Operator`, dunque, è possibile aggiungere nuove categorie di operatori al framework, ed implementando le interfacce specializzate vengono creati nuovi operatori facenti parte di tali categorie. Per tale motivo, il significato di fondo della gerarchia descritta è leggermente diverso da quelle utilizzate per rappresentare i processi di creazione e valutazione delle soluzioni e la loro rappresentazione.

4.2.4 L'Interfaccia `Algorithm`

L'ultimo elemento architetturale presente nella figura 4.1 è l'interfaccia che consente di implementare le proprie tecniche meta-euristiche qualora quelle presenti non risultassero sufficienti, ossia `Algorithm`. In particolare, essa entra in gioco una volta scelta la rappresentazione delle soluzioni, gli operatori da utilizzare e dopo aver definito le fasi di creazione e valutazione del problema. Anche in questo caso, l'interfaccia è molto semplice in quanto le segnature indicano che è necessario implementare due metodi, dove il primo esegue l'algoritmo, mentre il secondo restituisce un risultato generico, similmente a quanto avviene nell'interfaccia degli operatori. Tali segnature concedono grande libertà nel definire il proprio algoritmo, tuttavia,

nel framework sono state adottate determinate scelte progettuali per facilitare e guidare la fase implementativa.

In questa tesi non è stato necessario implementare una tecnica personalizzata, in quanto è stato utilizzato *NSGA-II*, algoritmo allo stato dell'arte, descritto nella sezione 3.6 e disponibile all'interno di jMetal. Tuttavia, l'implementazione disponibile di esso si basa su tali principi, perciò è utile chiarirli per comprendere meglio il comportamento del framework ed avere un punto di partenza per eventuali sviluppi futuri. Il punto chiave di tale scelta progettuale consiste nel fatto che molte tecniche meta-euristiche condividono un comportamento comune, che poi viene personalizzato da ciascuna tecnica. Tale comportamento comune, istanziato nel caso della tecnica degli *Algoritmi Genetici*, può essere riassunto con lo pseudocodice 3.1 (Sezione 3.1). Per tale motivo, un nuovo algoritmo appartenente a tale categoria potrebbe avere la necessità di implementare nuovamente solo l'operatore di Crossover, per esempio, oppure il metodo di valutazione delle soluzioni, senza alterare le parti restanti del *template*[2]. Tale caratteristica si può ottenere utilizzando un pattern di progettazione orientato agli oggetti chiamato *Template Method* e descritto in Gamma et al. [19, pp. 329-332], il diagramma è visibile nella figura 4.5.

Il pattern *Template Method*, dunque, viene caratterizzato una gerarchia la cui base è una classe astratta che specifica le segnature dei vari passi del template, mentre le classi concrete ne sovrascrivono, generalmente, solo alcuni. jMetal, dunque, agisce proprio in tal modo, fornendo una classe astratta per ogni categoria di tecnica meta-euristica da implementare. Per quanto riguarda quella degli *Algoritmi Genetici*, tale classe viene definita come `AbstractGeneticAlgorithm`, e contiene le segnature dei passi fondamentali indicati nello pseudocodice 3.1 (Sezione 3.1) ed implementa l'interfaccia `Algorithm` precedentemente descritta. Lo schema di tale classe è visibile nella figura 4.6. Per inserire un nuovo algoritmo nella struttura del framework, dunque, è sufficiente fornire un'implementazione concreta per la classe astratta che descrive la categoria a cui l'algoritmo stesso appartiene.

4.3 Conclusioni

Rimangono altre due brevi precisazioni da fare per definire completamente l'architettura di jMetal. La prima riguarda l'uso dei tipi di dato generici nelle interfacce

[2]Tale fenomeno viene chiamato *inversione del controllo*, ed è un principio che viene sfruttato nei framework per consentire all'utente finale di personalizzare il comportamento di una o più parti del framework stesso.

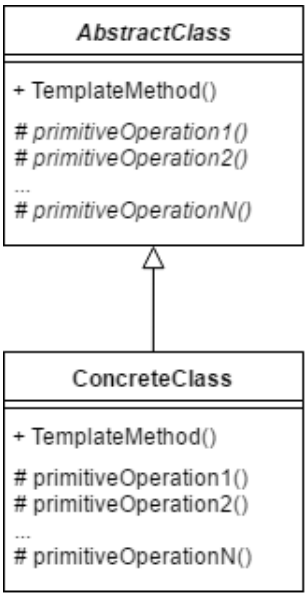


Figura 4.5: Schema del pattern di progettazione orientata agli oggetti chiamato *Template Method*.

e nelle classi descritte fino a questo punto. Il loro uso, infatti, consente di eliminare il concetto di tipologia a livello architetturale evitando, così, la creazione di classi `SolutionType`, `OperatorType` ecc. in quanto il tipo di dato generico veicola in se stesso tale concetto. Un ulteriore effetto positivo consiste nel fatto che tutti i componenti della tecnica meta-euristica sono allineati sullo stesso dato ed il compilatore impedisce il verificarsi di situazioni prive di senso. Per tale motivo, il blocco d'inizializzazione descritto nell'algoritmo 4.1 viene compilato con successo, contrariamente a quello dell'algoritmo 4.2.

La seconda precisazione riguarda le interfacce presenti nell'architettura nella figura 4.1 e descritte fino a questo punto. Tali interfacce, in Java come del resto in altri linguaggi di programmazione, specificano solamente le segnature dei metodi,

Algoritmo 4.1 Inizializzazione corretta di un algoritmo genetico.

1: $problem \leftarrow$ new $Problem\langle DoubleSolution\rangle$
2: $algorithm \leftarrow$ new $Algorithm\langle List\langle DoubleSolution\rangle\rangle$
3: $crossover \leftarrow$ new $CrossoverOperator\langle DoubleSolution\rangle$
4: $mutation \leftarrow$ new $MutationOperator\langle DoubleSolution\rangle$
5: $selection \leftarrow$ new $SelectionOperator\langle List\langle DoubleSolution\rangle, DoubleSolution\rangle$

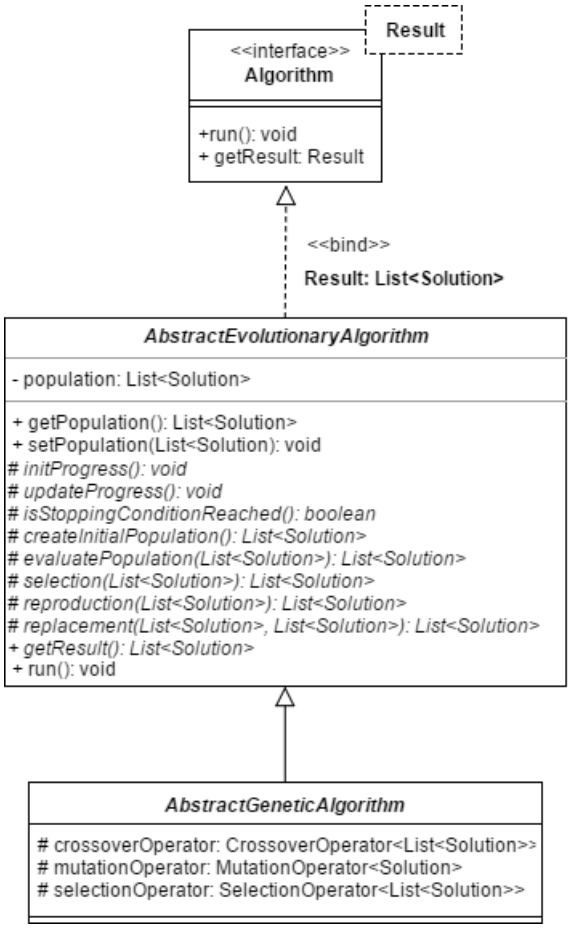


Figura 4.6: Diagramma delle classi astratte che rappresentano la tecnica metaeuristica degli *Algoritmi Genetici* all'interno di jMetal.

senza fornire un'implementazione. Per tale motivo, il framework fornisce una classe concreta per ciascuna di esse con la sola esclusione degli operatori, per i quali viene fornita un'interfaccia che estende quella di base, poiché essi dipendono strettamente dalla natura della soluzione. Tali classi vengono denominate anteponendo un `Default` oppure un `Abstract` al nome della rappresentazione scelta per, rispettivamente, soluzione e problema. Per cui, `DoubleSolution` viene implementata da `DefaultDoubleSolution` e `DoubleProblem` da `AbstractDoubleProblem`, ad esempio. Per quanto riguarda gli operatori, l'interfaccia estesa fornita dal framework assume il nome della tipologia di operatore gestita. Un operatore di *Crossover*, ad esempio, viene esteso da `CrossoverOperator`. Per gli algoritmi vengono composte le convenzioni utilizzate per operatori e problemi. Alla categoria a cui appartiene

Algoritmo 4.2 Inizializzazione errata di un algoritmo genetico.

1: *problem* ← new *Problem⟨IntegerSolution⟩*
2: *algorithm* ← new *Algorithm⟨List⟨BinarySolution⟩⟩*
3: *crossover* ← new *CrossoverOperator⟨DoubleSolution⟩*
4: *mutation* ← new *MutationOperator⟨DoubleSolution⟩*
5: *selection* ← new *SelectionOperator⟨List⟨DoubleSolution⟩, DoubleSolution⟩*

l'algoritmo viene anteposto un `Abstract` e la classe descrive la tipologia di algoritmo implementato.

Un dato algoritmo genetico, ad esempio, trova un'implementazione di base in `AbstractGeneticAlgorithm` la quale, come già precisato, segue le convenzioni del pattern *TemplateMethod*. Tali implementazioni consentono di ottenere i primi risultati velocemente, ma difficilmente sono sufficienti per soddisfare tutte le esigenze del decisore che tenta di risolvere un problema di ottimizzazione multi-obiettivo. Per tale motivo, una buona pratica che consente di integrare il framework all'interno del proprio software consiste nel creare nuove classi che estendano quelle contenenti le implementazioni concrete. Seguendo tale procedimento, si ottiene un'implementazione base di tutte le segnature indicate nelle interfacce in figura 4.1 ed è possibile sovrascriverle, quando necessario, o aggiungere i propri metodi ed attributi. Tale approccio è stato seguito nella fase di integrazione della nuova versione di *BestSub* con jMetal e tale fase viene descritta nel dettaglio nella sottosezione 6.3.4.

Nel capitolo 5, il quale apre la parte II, viene presentato l'obiettivo generale da raggiungere nell'ambito di questa tesi ed i quatto sotto-obiettivi in cui esso può essere diviso.

Parte II

NewBestSub: Implementazione e Valutazione

Capitolo 5

Obiettivi

Lo scopo di questo capitolo è descrivere più dettagliatamente gli obiettivi che questa tesi intende soddisfare. In particolare, nella sezione 5.1 viene descritto l'obiettivo generale e nelle relative sottosezioni vengono analizzate le quattro componenti in cui esso si divide.

5.1 Descrizione Dettagliata

L'obiettivo generale di questa tesi, come già descritto nella sezione 1.2, consiste nell'analisi di un particolare approccio alla riduzione del costo del processo di valutazione dei sistemi di IR. Tale approccio è stato studiato da Guiver et al. [20], Robertson [30] e Berto et al. [4] ed i risultati presentati nei relativi articoli sono stati ottenuti utilizzando un software originale chiamato *BestSub*. Tale obiettivo generale si divide in quattro sotto-obiettivi distinti, descritti nel seguito.

5.1.1 Implementazione di *NewBestSub*

Il funzionamento di *BestSub*, come si vede nella sezione 2.3, è basato su un'euristica caratterizzata da due limitazioni che possono avere un particolare effetto collaterale sulla stabilità dei risultati ottenuti ed incidere con decisione sull'efficienza del software stesso. In particolare, l'effetto sull'efficienza è piuttosto negativo in quanto impedisce di analizzare in un tempo ragionevole alcuni dataset caratterizzati da un numero di topic molto alto impedendo, di fatto, l'ottenimento di qualsiasi risultato. Il primo sotto-obiettivo da soddisfare, dunque, consiste nell'implementare una nuova versione di *BestSub*, chiamata *NewBestSub*, la quale sfrutti una rinnovata euristica con lo scopo di mettere in pratica l'approccio alla riduzione del costo del

processo di valutazione dei sistemi di IR studiato da Guiver et al. [20], Robertson [30] e Berto et al. [4], di evitare tale effetto collaterale sui risultati e di ottenere un'efficienza migliore. Tale euristica, in particolare, sfrutta la tecnica degli *Algoritmi Genetici* e viene approfondita nel capitolo 3. Al termine degli esperimenti svolti nell'ambito di questa tesi, inoltre, si intende rilasciare pubblicamente *NewBestSub* alla comunità dei ricercatori, i quali potranno così sfruttarlo per svolgere ulteriori esperimenti relativi alla riduzione del costo del processo di valutazione dei sistemi di IR mediante l'approccio studiato da Guiver et al. [20], Robertson [30] e Berto et al. [4].

5.1.2 Riproduzione dei Risultati Precedenti

Il sotto-obiettivo successivo da soddisfare consiste nello svolgimento di esperimenti analoghi a quelli svolti da Guiver et al. [20], Robertson [30] e Berto et al. [4] utilizzando *NewBestSub*, al fine di verificare se esso è in grado di riprodurre i risultati originali ottenuti con *BestSub*, i quali vengono descritti nella sezione 2.2. In tal modo, è possibile certificarne l'efficacia e, conseguentemente, verificare se la bontà di tali risultati originali è indipendente o meno dal software utilizzato. Inoltre, si intende analizzare anche i miglioramenti ottenuti dal punto di vista dell'efficienza della computazione mediante l'utilizzo della nuova euristica e discutere i vantaggi rispetto a quella precedente. La riproduzione dei risultati originali è particolarmente rilevante poiché essi sono stati originariamente ottenuti da un singolo gruppo di ricerca il quale ha utilizzato un software specifico e sviluppato ad hoc, mai reso ufficialmente disponibile. Tali risultati, inoltre, sono stati pubblicati su periodici importanti. In particolare, l'articolo di Guiver et al. [20] è stato pubblicato sull'*ACM TOIS*[1], mentre quelli di Berto et al. [4] e Robertson [30] sono stati pubblicati nell'ambito dell'*ICTIR* e dell'*ECIR*, due congressi importanti per l'*Information Retrieval*.

5.1.3 Generalizzazione dei Risultati

Il sotto-obiettivo da soddisfare dopo aver verificato l'efficacia e l'efficienza di *NewBestSub* ed averne certificato il funzionamento consiste nello svolgimento di esperimenti per generalizzare e consolidare i risultati originali andando ad analizzare diversi aspetti importanti per il processo di riduzione del costo della valutazione dei sistemi di IR e l'effetto dato dall'uso di dataset più recenti di quelli analizzati da Guiver et al. [20], Robertson [30] e Berto et al. [4] e di dataset derivanti

[1] `https://tois.acm.org/`

dall'applicazione della tecnica del pooling. Gli esperimenti svolti per soddisfare questo sotto-obiettivo, inoltre, consentono di verificare il superamento della limitazione che affligge l'euristica sfruttata da *BestSub*, legata alla stabilità dei risultati stessi, da parte di quella nuova sulla quale viene basato il funzionamento dell'algoritmo genetico *NSGA-II*, utilizzato da *NewBestSub*.

5.1.4 Espansione dei Risultati

Il processo di valutazione dei sistemi di IR, come descritto nella sezione 2.1, richiede l'intervento di un valutatore umano che assegna dei giudizi, per essere portato a termine. l'ultimo sotto-obiettivo da soddisfare mira ad espandere i risultati ottenuti durante le fasi di riproduzione e generalizzazione svolgendo nuovi esperimenti che espandano gli studi di Guiver et al. [20], Robertson [30] e Berto et al. [4]. In particolare, tali esperimenti mirano ad analizzare gli effetti di due particolari metodologie, ossia la strategia di selezione a priori dei topic sviluppata da Soboroff et al. [34] e l'analisi dei punteggi di hubness proposta da Mizzaro et al. [25] al fine di ottenere delle nuove metodologie pratiche per selezionare sottoinsiemi di topic da utilizzare per il processo di riduzione del costo della valutazione dei sistemi di IR. In altre parole, si intende ottenere delle metodologie che permettano di valutare i sistemi di IR utilizzando meno topic ed evitando di chiamare in causa valutatori umani.

Più dettagliatamente, i sistemi di *IR* vengono valutati utilizzando i sottoinsiemi di topic individuati mediante le due metodologie analizzate e quelli composti selezionando i topic stessi casualmente. Dall'esito di tale valutazione, dunque, si intende ottenere le nuove metodologie pratiche precedentemente descritte.

Nel capitolo 6 viene descritta la nuova versione di *BestSub* chiamata *NewBestSub*, la quale sfrutta *jMetal* per implementare il nuovo processo di riduzione dei costi della valutazione dei sistemi di IR mediante algoritmi genetici descritto nella sezione 3.7. In particolare, viene analizzata l'architettura del software stesso e come viene concretamente svolta l'attività di integrazione con il framework. Inoltre, vengono presentate tutte le funzionalità utilizzabili da un utente esterno e vi è una breve discussione sulle tecnologie utilizzate.

Capitolo 6

Architettura di *NewBestSub*

Lo scopo di questo capitolo è descrivere l'architettura della nuova versione di *BestSub*, chiamata *NewBestSub*. In particolare, nella sezione 6.1 viene analizzato il pattern di progettazione sul quale tale nuova versione si basa. Successivamente, nella sezione 6.2 vi è una descrizione generale delle varie componenti in cui *NewBestSub* si divide, mentre nella sezione 6.3 vi è un'analisi più dettagliata di tali componenti architetturali ed, inoltre, viene descritto il lavoro svolto per ottenere l'integrazione del software con *jMetal*. Infine, nella sezione 6.4 vengono tratte le conclusioni riguardanti l'architettura di *NewBestSub* nella sua interezza.

6.1 Il Pattern MVC

Una delle prime decisioni prese durante la fase d'implementazione di *NewBestSub* è stata quella di affidarsi ad un pattern di progettazione architetturale per poter separare in maniera netta la logica di controllo del programma dalla presentazione dei dati e dalla logica di business ed ottenere, in tal modo, uno schema architetturale efficace fin dai primi istanti di tale fase. Tutto ciò viene reso possibile dall'adozione di un pattern di progettazione architetturale chiamato *Model-View-Controller*, spesso abbreviato come *MVC*. Una rappresentazione intuitiva della struttura che tale pattern permette di ottenere è visibile nella figura 6.1. Esso si compone di tre distinte entità, chiamate *Controller*, *Model* e *View*. Tali entità hanno il compito di, rispettivamente, gestire la logica di controllo, incapsulare la logica di business ed implementare le funzionalità di presentazione dei dati.

Il senso di tale struttura è che il *Controller* ha accesso diretto al *Model* ed alla *View* e da quest'ultima riceve, generalmente, l'input dall'utente. A seconda del-

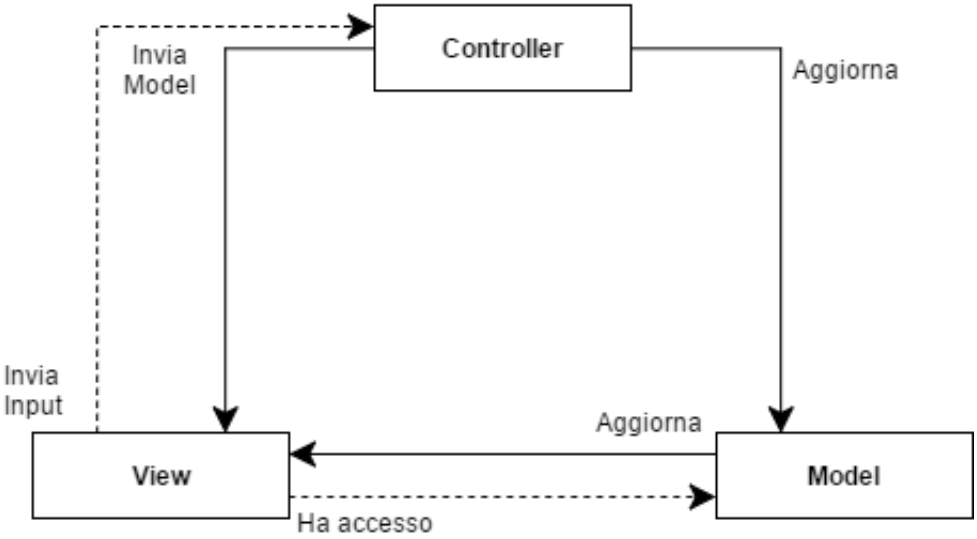


Figura 6.1: Schema intuitivo della struttura del pattern MVC.

l'input ricevuto, successivamente, il controller stesso aggiorna lo stato del *Model* sfruttando i metodi di quest'ultimo. Il *Controller*, infine, invia il *Model* aggiornato alla *View*, la quale interroga il *Model* stesso per ottenere e visualizzare i risultati dell'elaborazione. Un software generico può contenere al suo interno più di un *Controller*[1], dove ognuno di essi può gestire diverse istanze di un *Model*. Infine, vi possono essere più *View* con il compito di visualizzare lo stato di una precisa tipologia di *Model*. La scelta di adottare il pattern *MVC* e la separazione che esso permette di ottenere si è rivelata vantaggiosa in diversi momenti durante l'attività implementativa, alcuni dei quali vengono approfonditi nella sottosezione 7.1.5.

6.2 Descrizione Generale

A questo punto è possibile aumentare il livello del dettaglio nell'analisi dell'architettura del software, osservando le quattro componenti in cui esso si divide nel diagramma di package visibile nella figura 6.2. Tale diagramma è utile per descrivere in maniera più approfondita qual è il ruolo svolto nel sistema dalle singole componenti, mentre la struttura interna di ciascuna di esse viene descritta nel seguito. I package visualizzati nello schema sono contenuti all'interno di un ulteriore package definito dal nome completamente qualificato `it.uniud.newbestsub`. Il nome com-

[1]Nei framework MVC dedicati allo sviluppo di applicazione web, ad esempio, è una pratica comune quella di avere un numero di *Controller* pari al numero di entità del dominio applicativo.

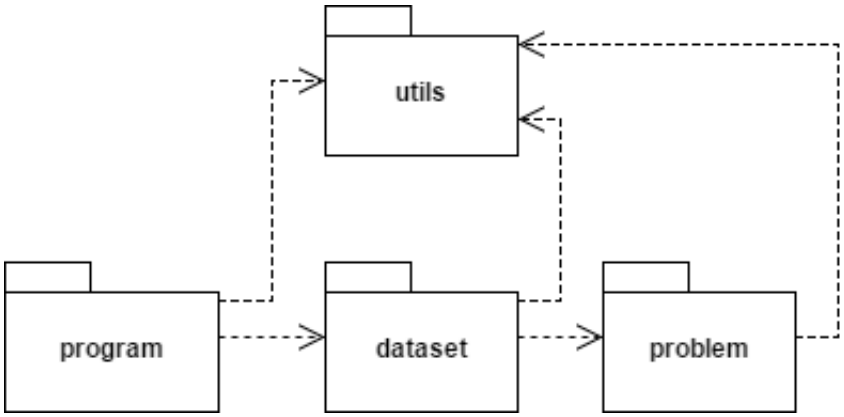


Figura 6.2: Diagramma di package che descrive le quattro componenti in cui *NewBestSub* viene diviso.

pletamente qualificato di una classe contenuta all'interno del package `dataset`, ad esempio, è `it.uniud.newbestsub.dataset.NomeClasse`.

6.2.1 Il Package `utils`

Il package `utils` ha il compito di fornire strumenti utili al resto del sistema. Al suo interno vi sono formule matematiche generiche, costanti comuni e metodi per accedere alle funzionalità di logging. Come si può notare osservando il diagramma, tutti gli altri package dipendono da esso, soprattutto per le funzionalità di logging stesse e per i valori contenuti in alcune delle costanti, mentre il resto delle funzionalità viene sfruttato principalmente dal package `problem`.

6.2.2 Il Package `program`

All'interno del package `program` avviene la gestione dell'interazione con l'utente. Il modo con il quale un utente può avviare e regolare l'esecuzione di *NewBestSub*, infatti, consiste nell'uso di un insieme di opzioni inviate da riga di comando. All'interno di tale package, dunque, avviene l'attività di parsing dei valori ricevuti per ciascuna delle opzioni e la gestione del flusso d'esecuzione sulla base di tali valori.

6.2.3 Il Package `dataset`

All'interno del package `dataset` è presente l'implementazione del pattern *MVC* descritto nella figura 6.1. Il controller, in particolare, prende in carico i parametri d'esecuzione analizzati all'interno del package `program` ed aggiorna lo stato di una

o più istanze del modello sfruttando i metodi forniti dalla classe che implementa il modello stesso. Tale operazione comporta il caricamento del dataset in input, l'esecuzione di uno o più esperimenti scelti e tarati con i parametri forniti dall'utente ed il salvataggio dei risultati ottenuti nelle strutture dati interne al modello. Infine, il controller recupera e fornisce alla vista il risultato disponibile nello stato del modello stesso, la quale si occupa di stampare il tutto in un formato utilizzabile dall'utente.

Le operazioni svolte dal controller, in realtà, possono essere leggermente più ampie rispetto a quanto descritto poiché esso può eseguire alcune elaborazioni sul dataset in input e sul risultato ottenuto, ovvero prima e dopo l'interazione con il modello. Tali elaborazioni, dunque, consentono di rendere disponibili all'utente alcune funzionalità aggiuntive, la cui natura viene descritta nella sottosezione 6.3.3, mentre il modo in cui esse vengono utilizzate in termini di processo viene approfondito nella sottosezione 7.1.5.

6.2.4 Il Package `problem`

Il package `problem` è il punto in cui avviene l'integrazione di *NewBestSub* con *jMetal*. Al suo interno vengono concretamente implementate e legate al software le entità architetturali del framework, indicate nella figura 4.1. Le classi contenute all'interno del package vengono istanziate solamente all'interno del modello, poiché è grazie ad esse che gli esperimenti configurati dal controller attraverso i parametri inviati dall'utente possono effettivamente svolgersi. Il modello stesso, dunque, codifica il processo generale degli esperimenti, e delega compiti specifici alle classi contenute nel package stesso.

6.3 Descrizione Dettagliata

Avendo fissato la descrizione generale dei ruoli delle quattro componenti nelle sottosezioni precedenti, è possibile aumentare ulteriormente il livello di dettaglio osservando quali sono le classi effettivamente contenute nei package e quali sono i loro metodi ed attributi.

6.3.1 Il Package `utils`

Per quanto riguarda il package `utils`, vi sono due classi statiche ed indipendenti, visibili nel diagramma in figura 6.3[2]. La classe `Tools` è un toolkit di formule utili per il dominio applicativo e consente, in particolare, di calcolare la media ponderata per un array inviato come parametro e la *distanza di Hamming* tra due stringhe di uguale lunghezza. Tale formula va intesa come strumento per poter confrontare strutturalmente tali stringhe e viene descritta nella definizione intuitiva 6.1, per la quale:

Definizione 6.1. La *distanza di Hamming* tra due stringhe di lunghezza uguale è il numero di posizioni nelle quali i simboli corrispondenti sono diversi.

Ad esempio, si immagini di definire una stringa x ed una stringa y, con $x = 1011101$ e $y = 1001001$. la chiamata `stringComparison(x,y)` restituisce, come risultato, un valore pari a 2. L'ultimo metodo presente nella classe `Tools`, infine, consente di accedere alle funzionalità di logging e viene utilizzato estensivamente nell'implementazione di *NewBestSub*, poiché essa non ha alcuna forma di interfaccia grafica. Per tale motivo, risulta utile stampare dei messaggi che mantengano informato l'utente sullo stato di avanzamento dell'elaborazione.

La classe `Constants` è un contenitore di costanti utili per il funzionamento di *NewBestSub*. Non è necessario entrare nel dettaglio di ogni singola costante, tuttavia, osservandone i nomi, è facile notare come la maggior parte di esse abbia a che fare con la gestione di path, nomi ed estensioni dei file che vengono manipolati dal sistema.

6.3.2 Il Package `program`

Il package `program` contiene, al suo interno, un'unica classe, il cui diagramma è visibile nella figura 6.4. Tale classe rappresenta il punto d'entrata di *NewBestSub* per l'utente esterno e contiene il metodo `main`, al quale vengono passate come parametro le opzioni da riga di comando indicate dall'utente stesso. Il secondo metodo riportato nello schema si occupa di generare tutte le opzioni utilizzabili dall'utente con la relativa descrizione. All'interno del metodo `main`, dunque, viene effettuato il parsing delle opzioni effettivamente indicate dall'utente stesso ed i loro valori vengono salvati in apposite variabili. Sulla base di tali valori viene istanziato il controller ed avviata l'esecuzione degli esperimenti scelti. Infine, eventuali eccezioni lanciate dalle altre componenti vengono catturare e gestite, stampandone il relativo messaggio d'errore.

[2] Nello schema non vengono riportati i valori delle costanti per non compromettere la leggibilità, poiché alcuni di essi sono stringhe piuttosto lunghe.

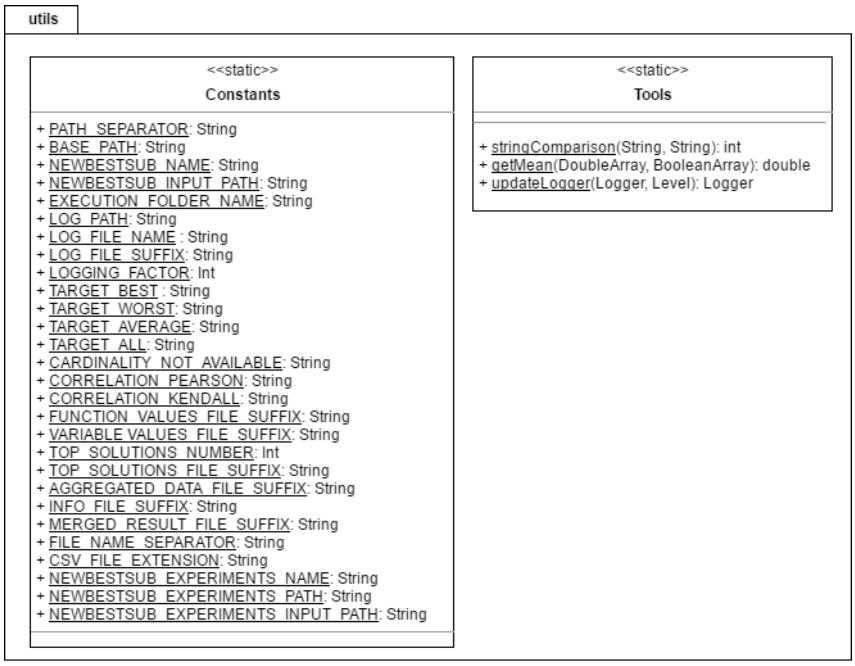


Figura 6.3: Diagramma delle classi contenute nel package `utils`.

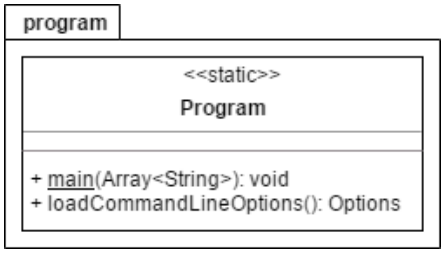


Figura 6.4: Diagramma della classe contenuta nel package `program`.

6.3.3 Il Package `dataset`

Per quanto riguarda il package `dataset`, che coordina l'esecuzione degli esperimenti sfruttando l'implementazione del pattern *MVC*, è necessario fare una breve puntualizzazione, prima di procedere con l'attività di analisi. Nel diagramma di tale package, visibile nella figura 6.5, per alcune delle classi contenute al suo interno non vengono riportati tutti gli attributi ed i metodi. Allo stesso tempo, risultano tutti necessari per descrivere l'architettura del package efficacemente e senza ambiguità. Per ottenere un compromesso tra la chiarezza del diagramma e la completezza della descrizione, dunque, non tutti gli attributi e metodi nominati nel seguito si possono effettivamente vedere nel diagramma stesso, nonostante siano comunque presenti

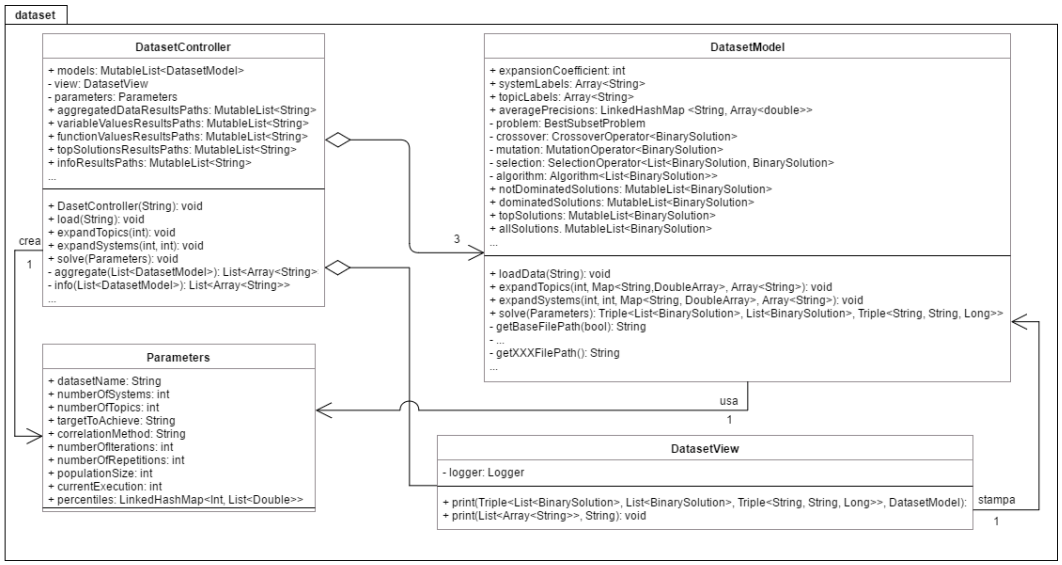


Figura 6.5: Diagramma delle classi contenute nel package `dataset`.

nell'implementazione della classe di appartenenza.

6.3.3.1 Il Controller

La prima classe ad essere chiamata in causa è `DatasetController`. Come illustrato nel diagramma visibile nella figura 6.1, il suo compito è aggiornare lo stato del modello utilizzando le funzioni che quest'ultimo fornisce, sulla base delle opzioni da riga di comando indicate in input dall'utente. Una volta fatto ciò, il modello stesso viene inviato alla vista, la quale ha il compito di aggregare e fornire in output i risultati ottenuti. Il controller viene definito nel diagramma come un aggregato di tre istanze del modello e di una singola istanza della vista. Le tre istanze del modello vengono immagazzinate in un'apposita struttura dati e rappresentano gli esperimenti *Best*, *Worst* e *Average* delineati nella sezione 2.2. Le istanze del modello stesso presenti nel controller, dunque, sono esattamente tre, una per esperimento. In tal modo, i risultati per ciascun esperimento vengono adeguatamente divisi e non vi sono rischi di ambiguità nelle successive manipolazioni degli stessi. Si tratta di uno dei vantaggi che il pattern MVC permette di ottenere. I rimanenti attributi rappresentano informazioni utili per la computazione quali il path per il dataset in input e gli esperimenti scelti dall'utente. Infine, vi è un riferimento all'oggetto che permette di accedere alle funzionalità di logging.

A questo punto, prima di descrivere i metodi del controller, è necessario fare

una seconda digressione descrivendo la classe `Parameters`, collegata all'attributo omonimo presente nel controller stesso. Essa, in particolare, non ha alcun metodo e rappresenta un semplice contenitore di dati. La sua presenza nell'architettura è dovuta al fatto che il controller deve poter fornire, in qualche modo, i parametri inviati dall'utente per regolare l'esecuzione dell'esperimento all'istanza corrente del modello. La soluzione più semplice per ottenere il tutto consiste nel passare tali parametri attraverso l'interfaccia del metodo `solve` del controller, che ha il compito di avviare ed eseguire concretamente l'esperimento scelto, chiamando l'omonimo metodo del modello. Tuttavia, i parametri sono molti e ciò appesantisce l'interfaccia stessa, rendendola poco leggibile. Inoltre, alcuni parametri possono essere opzionali, e ciò può portare ad avere chiamate a tale metodo con argomenti nulli. Per risolvere tale problema, è stato applicato un refactoring alla prima versione del metodo chiamato *Introduce Parameter Object*, che consiste nel creare una nuova classe che funga da contenitore per i parametri da inviare al modello. Il controller, dunque, istanzia un solo oggetto di tale classe e configura i valori dei parametri ricevuti all'interno di esso il quale, successivamente, viene inviato al modello attraverso l'interfaccia del metodo `solve`. Così facendo, l'interfaccia stessa richiede la presenza di soli due argomenti, risultando decisamente più leggera ed evitando l'invio di valori nulli. Il modello, infine, estrae i valori dei parametri dall'oggetto stesso e li mappa nei propri attributi. Gli attributi contenuti nel diagramma della classe `Parameters`, dunque, sono quelli che vengono incapsulati all'interno di tale oggetto e successivamente estratti all'interno del modello. Per tale motivo, i diagrammi di controller e modello presenti nella figura 6.5 hanno una freccia indicante la navigabilità verso una singola istanza della classe `Parameters`. La differenza sta nel fatto che il primo la crea, mentre il secondo la sfrutta concretamente.

Riprendendo la descrizione del controller, rimangono da analizzare alcuni metodi metodi legati all'esecuzione standard del software. Tali metodi, in particolare, si occupano di caricare in uno o più modelli il dataset in input, a seconda del numero di esperimenti scelti, e di avviare ciascun esperimento con i parametri indicati dall'utente. Inoltre, vi sono metodi che aggregano i risultati ottenuti in un'unica struttura da inviare alla vista per l'output, i quali producono dei file con informazioni legate allo svolgimento dell'esecuzione, quali la durata temporale della stessa ed altro. Negli ultimi due casi, le operazioni si svolgono una volta terminata l'esecuzione dell'esperimento, analizzando la struttura dati che contiene i modelli aggiornati da aggregare e dai quali estrarre i dati. Il risultato di tali operazioni, successivamente, viene passato alla vista per la stampa.

Le funzionalità aggiuntive che il controller può svolgere vengono rese possibili da alcuni attributi e metodi contenuti nel diagramma in figura 6.5. Tali funzionalità aggiuntive sono tre e consistono, in particolare, nell'espansione del dataset in input con topic o sistemi finti e nella fusione dei risultati di più esecuzioni di un dato esperimento. Le prime due funzionalità aggiuntive permettono di testare l'efficienza del software. La possibilità di espandere il dataset di input lungo due dimensioni, infatti, permette di verificare fino a che punto l'esecuzione del software stesso rimane sostenibile. L'ultima funzionalità aggiuntiva, invece, consente di ottenere un risultato complessivo migliore aggregando quelli di più esecuzioni successive. Alcuni attributi, in particolare, servono a tracciare i file prodotti in output da ciascuna di tali esecuzioni. Vi è poi un metodo per avviare ciascuna di tali funzionalità, mentre gli ultimi due ad essere indicati nel diagramma servono a ripulire il disco rigido dai risultati delle singole esecuzioni, una volta terminata la loro fusione in un'unica struttura dati.

6.3.3.2 Il Modello

Chiarito il ruolo del controller, è possibile passare alla descrizione del modello, implementato dalla classe `DatasetModel` nel diagramma in figura 6.5. Prima di poter far ciò, tuttavia, è necessaria ancora una precisazione, per chiarire una scelta legata ai nomi assegnati ai metodi. Come si può notare negli schemi di modello e controller, vi sono alcuni metodi che hanno lo stesso nome ma parametri diversi. Tale scelta di nomi è legata al fatto che il controller è un aggregato di tre diverse istanze del modello e spesso è necessario eseguire delle operazioni preparatorie per ciascuna di tali istanze prima di sfruttare concretamente le funzionalità fornite da tali metodi. Perciò, nel metodo all'interno del controller è presente un ciclo dove vengono eseguite tali operazioni preparatorie su ogni modello attivato nell'ambito dell'esecuzione corrente. Tale ciclo viene seguito da una chiamata, anche in questo caso per ciascuno dei modelli attivi, al metodo omonimo presente in `DatasetModel`. In tal modo il metodo corrispondente riceve, come parametri, i dati preparati dal controller i quali vengono sfruttati nell'esecuzione della funzionalità che incapsula. In tal modo, ciascun modello può continuare ad essere indipendente da quelli rimanenti, rispettando la struttura imposta dal pattern *MVC*. Per chiarire con un esempio questa descrizione, si supponga di voler espandere il dataset in input un dato numero di nuovi sistemi generati randomicamente. Per svolgere tali operazioni vengono chiamati in causa i due metodi `expandSystems`. Lo pseudocodice 6.1 descrive cosa avviene nel metodo del controller, mentre lo pseudocodice 6.2 è per

Algoritmo 6.1 Pseudocodice per il metodo `expandSystems` (controller).

```
1: systemLabels ← GenerateRandomLabels(coefficient)
2: randomizedAveragePrecisions ← new Map
3: while systemLabels.hasNext() do
4:     randomizedAveragePrecisions[systemLabels.next] ← RandomArray()
5: end while
6: while models.hasNext() do
7:     model ← models.next()
8:     c ← coefficient
9:     t ← trueNumberOfSystems
10:    r ← randomizedAveragePrecisions
11:    s ← systemLabels
12:    model.expandSystems(c, t, r, s)
13: end while
```

quello del modello. Come si può vedere, il controller ha il compito di generare i nuovi vettori casuali di AP, mentre il modello riceve tali vettori e li aggiunge all'interno della relativa struttura dati.

Ricapitolando quanto detto fino ad ora sul modello, un'istanza di esso rappresenta uno dei tre esperimenti che possono essere svolti, la classe `DatasetModel` incapsula la logica di business necessaria per la loro esecuzione ed in alcuni attributi vengono immagazzinati i parametri inviati dal controller per regolare l'esecuzione dell'esperimento rappresentato. Inoltre, ve ne sono altri i quali consentono lo svolgimento di alcune delle funzionalità aggiuntive già citate. Infine, ve ne sono di ulteriori i quali rappresentano istanze degli elementi architetturali di *jMetal* integrati con *NewBestSub*. Tutti gli attributi elencati, esclusi quelli legati alle funzionalità aggiuntive, partecipano ad ogni esecuzione del software.

Per quanto riguarda i metodi della classe `DatasetModel`, uno dei primi ad essere chiamato in causa è quello per il caricamento del dataset in input. Tale metodo effettua il parsing del dataset stesso e popola alcuni attributi, immagazzinando in apposite strutture dati le etichette identificative dei sistemi e dei topic contenuti nel dataset stesso ed i relativi valori di AP. Infine, si occupa di calcolare ed archiviare nel rispettivo attributo le medie dei vettori di AP così ottenuti. In sostanza, all'interno del metodo vengono ottenute e memorizzate tutte le informazioni contenute nella matrice in figura 2.4.

Una volta completato il caricamento dei dati, viene chiamato il metodo che av-

Algoritmo 6.2 Pseudocodice per il metodo `expandSystems` presente (modello).

```
1:  newNumberOfSystems ← numberOfSystems + expansionCoefficient
2:  if newNumberOfSystems < trueNumberOfSystems then
3:      averagePrecisions ← originalAveragePrecisions
4:      systemsToKeep ← averagePrecisions.entries.TAKE(newNumberOfSystems)
5:      averagePrecisions ← new LinkedMap
6:      while systemsToKeep.HASNEXT() do
7:          aSystemToKeep ← systemsToKeep.NEXT()
8:          averagePrecisions.PUT(aSystemToKeep.key, aSystemToKeep.value)
9:      end while
10: else
11:     while randomizedSystemLabels.HASNEXT() do
12:         randSysLab ← randomizedSystemLabels.NEXT()
13:         if randomizedAveragePrecisions[randSysLab].TOLIST() is not null then
14:             b ← randomizedAveragePrecisions[randSysLab].TOLIST()
15:             averagePrecisions.PUT(randomizedSystemLabel, b)
16:         else
17:             averagePrecisions.PUT(EMPTYLIST.TOTYPEDARRAY(), b)
18:         end if
19:     end while
20: end if
```

via concretamente l'esecuzione dell'esperimento. Tale metodo riceve i parametri inviati dal controller ed incapsulati in un'istanza della classe `Parameters` attraverso la propria interfaccia e, come prima cosa, determina quale dei tre esperimenti ha il compito di eseguire. A seconda dell'esito di tale controllo, l'esecuzione può procedere secondo due binari. Se l'esperimento scelto è di tipo *Average*, viene svolto direttamente all'interno del metodo stesso poiché è un problema facile, che non necessita di essere modellato come un problema di ottimizzazione multi-obiettivo sfruttando *jMetal*. Se è necessario svolgere un esperimento di tipo *Best* o *Worst*, il metodo istanzia le già citate entità architetturali del framework con i parametri ricevuti e lancia la risoluzione dell'esperimento modellato come problema di ottimizzazione multi-obiettivo, la quale viene svolta utilizzando l'algoritmo genetico *NSGA-II*, descritto nella sezione 3.6. In entrambi i casi, una volta terminata l'esecuzione dell'algoritmo stesso è necessario immagazzinarne il risultato in apposite strutture dati le quali, nel diagramma visibile nella figura 6.5, corrispondono agli attributi deno-

minati `xxxxSolutions`. Prima di procedere con il resto della descrizione di quanto svolto dal metodo, è necessario chiarire una questione che porta all'avere ben quattro attributi i quali assumono tale denominazione, derivanti proprio dalla natura di problema di ottimizzazione multi-obiettivo degli esperimenti *Best* e *Worst*.

Come viene spiegato nella sezione 3.7, per un decisore che ha come scopo la risoluzione di un problema multi-obiettivo, le decisioni da prendere sono quelle *non dominate* perché, intuitivamente, minimizzano (massimizzano) maggiormente i valori calcolati dalle funzioni obiettivo per ciascun criterio di scelta. Ci si focalizzi, a questo punto, sul processo di riduzione del costo della valutazione dei sistemi di IR studiato da Guiver et al. [20], Robertson [29] e Berto et al. [4] e descritto nella sezione 2.2. Tale processo consiste, concretamente, nell'analizzare i sottoinsiemi di topic che *per ogni cardinalità* massimizzano (nel caso *Best*) o minimizzano (nel caso *Worst*) i valori di correlazione. Nella modellazione di tali esperimenti le funzioni obiettivo sono due e consistono, dunque, nei valori di cardinalità e correlazione del sottoinsieme analizzato. Tali problemi, successivamente, vengono concretamente rappresentati e risolti grazie a *jMetal* e quello che le entità del framework restituiscono, dunque, è un *insieme di soluzioni non dominate*, rispettando così il significato veicolato da un problema di ottimizzazione multi-obiettivo. Questo aspetto porta ad avere i quattro attributi citati nel paragrafo precedente. Può capitare, infatti, che l'insieme di soluzioni *non dominate* restituito dalle entità di *jMetal* non comprenda *tutte* le possibili cardinalità, perché per quelle mancanti sono state analizzate solamente soluzioni *dominate*. Per tale motivo, durante la fase di valutazione delle soluzioni stesse, viene immagazzinata in `dominatedSolutions` la miglior soluzione *dominata* per ogni cardinalità, mentre in `nonDominatedSolutions` vengono archiviate tutte le soluzioni *non dominate* ottenute una volta terminata la fase di valutazione stessa. Tali contenitori di soluzioni vengono poi fusi in `allSolutions` dove, per ogni cardinalità, viene selezionata la miglior soluzione *non dominata*, se disponibile; in caso contrario, la migliore soluzione dominata. Il caso ideale, naturalmente, si ha quando `dominatedSolutions` è una struttura dati vuota, ma ciò difficilmente si verifica. Ad ogni modo, con questa procedura si ottiene la miglior configurazione disponibile di sottoinsiemi di topic per soddisfare gli obiettivi dell'esperimento corrente, indipendentemente dal fatto che tale esperimento corrisponda a quello di tipo *Best* o *Worst*.

Nell'esperimento *Average* le operazioni precedentemente descritte non sono necessarie in quanto durante il suo svolgimento viene generata esattamente una soluzione casuale per ciascuna cardinalità. Ciascuna soluzione, dunque, immagazzi-

nata all'interno di `nonDominatedSolutions` ed `allSolutions` consiste, semplicemente, in una copia di tale struttura. Vi è, infine, un ultimo attributo nella forma `xxxxSolutions` non ancora descritto, chiamato `topSolutions`, il quale viene sfruttato dal modello solamente nell'ambito degli esperimenti *Best* e *Worst*. Durante la fase di valutazione delle soluzioni, oltre al risultato delle operazioni descritte fino a questo punto, vengono immagazzinate in tale attributo le dieci migliori soluzioni per ciascuna cardinalità. Tali insiemi di soluzioni, in particolare, possono essere costituiti da una soluzione *non dominata* unita a nove soluzioni *dominate*, oppure da dieci soluzioni *dominate*. In realtà, il software non è in grado di garantire che ve ne siano esattamente dieci per ogni cardinalità poiché, intuitivamente, jMetal tende a bloccare l'esplorazione dello spazio di ricerca lungo una data direzione quando trova una soluzione *non dominata*. Tale valore, dunque, è una limitazione superiore e non necessariamente esatta.

Una volta terminata la fase di valutazione ed organizzazione delle soluzioni nelle strutture descritte nel paragrafo precedente il metodo di risoluzione esegue un'ultima operazione prima di terminare, che consiste nel definire la distribuzione dei topic del dataset all'interno dei sottoinsiemi che caratterizzano ciascuna soluzione. Per calcolare tale distribuzione, inizialmente viene creata una matrice cardinalità per topic. Successivamente, viene eseguita una scansione di tutte le soluzioni presenti in `allSolutions`. Per ciascuna soluzione, si analizza ogni topic del relativo sottoinsieme e si scrive una `B` (`W`) nella relativa cella, se l'esperimento è di tipo *Best* (*Worst*), identificata all'interno della matrice dall'etichetta del topic analizzato e dalla cardinalità del sottoinsieme della soluzione stessa.

Per quanto riguarda i rimanenti metodi non legati alle funzionalità aggiuntive, essi vengono utilizzati dal controller come funzioni d'utilità per agire sui risultati immagazzinati nel modello e consistono, ad esempio, nell'ordinare le soluzioni per cardinalità, ottenere il valore di correlazione per una soluzione archiviata in `allSolutions` di cardinalità data e quant'altro. Infine, i metodi nella forma `getXXXFilePath` servono ad ottenere i percorsi dei file all'interno dei quali vengono scritte dalla vista tutte le informazioni prodotte dal modello come, ad esempio, i valori delle funzioni obiettivo ed i sottoinsiemi di topic che caratterizzano le soluzioni.

Gli ultimi elementi da trattare nella descrizione del modello sono quelli legati alle funzionalità aggiuntive, presenti nello schema in figura 6.5, i quali consentono di effettuare l'espansione del dataset in input aggiungendo topic (sistemi) finti. Tali metodi, in particolare, ricevono come parametri tali topic (sistemi) finti da aggiungere dal controller, il quale ha il compito di generarli, ed eseguono concretamente

l'espansione del dataset stesso, archiviato nello stato del modello.

6.3.3.3 La Vista

A questo punto è possibile procedere con la descrizione della vista, implementata dalla classe `DatasetView` visibile nel diagramma presente nella figura 6.5. Rispetto a quella di controller e modello, la sua interfaccia è piuttosto semplice poiché viene caratterizzata solamente da due metodi e da un riferimento al logger. Nello diagramma, inoltre, si vede come essa abbia un accesso diretto ad un'istanza del modello, che interroga per ottenere le informazioni da stampare. Il controller, dunque, rende disponibile ciascuno dei modelli attivi alla vista stessa una volta aggiornatone lo stato. Osservando le segnature dei due metodi nello schema si può notare come, in realtà, esse non contengano solamente un parametro di tipo `DatasetModel`. Questo avviene perché il controller invia alla vista i dati già estratti dal modello corrente; si tratta di una semplice facilitazione nell'implementazione che non pregiudica l'utilizzo del pattern MVC.

Ad ogni modo, il primo metodo ha il compito di stampare il risultato complessivo dell'esecuzione di un singolo esperimento. In particolare, esso riceve come parametro una tripla, i cui elementi sono le soluzioni finali, le migliori dieci per ogni cardinalità[3] ed informazioni legate all'esecuzione del software, quali tempo totale impiegato per raggiungere la terminazione e nome del thread utilizzato. Inoltre, viene inviato anche un riferimento all'istanza del modello, la quale viene interrogata per ottenere i percorsi in cui creare i file contenenti le varie componenti del risultato complessivo. Tali componenti, in particolare, sono i valori delle funzioni obiettivo per ciascuna soluzione, la composizione dei sottoinsiemi di topic che le caratterizzano e le migliori dieci per ciascuna cardinalità. Le informazioni legate all'esecuzione concreta, invece, vengono utilizzate per finalità di logging.

Il secondo metodo disponibile è più generico poiché viene utilizzato dal controller per stampare l'esito di operazioni eseguite su tutti i modelli attivi. Una di queste operazioni, ad esempio, consiste nell'aggregazione di ciascun risultato complessivo in un'unica struttura al fine di ottenere informazioni aggiuntive quali, ad esempio, la distribuzione globale dei topic negli esperimenti scelti.

[3] Gli attributi `allSolutions` e `topSolutions` del modello.

6.3.4 Il Package `problem`

La descrizione della vista esaurisce l'analisi del package `dataset` ed è possibile, a questo punto, analizzare il package `problem`, l'ultimo a non essere ancora descritto tra quelli presenti nel diagramma visibile nella figura 6.2. Al suo interno viene svolta l'integrazione di *NewBestSub* con l'architettura di *jMetal*, secondo l'approccio descritto negli ultimi paragrafi della sezione 4.2. Riassumendo la descrizione di tale approccio, il framework fornisce un insieme di classi con un'implementazione di base di ciascuna entità architetturale presente nel diagramma visibile nella figura 4.1, con la sola esclusione degli operatori, per i quali vengono solamente definite interfacce che estendono quelle di base, poiché l'implementazione delle loro funzionalità è strettamente connessa alla rappresentazione delle soluzioni. La pratica migliore per ottenere tale integrazione, dunque, consiste nell'estendere le classi concrete per poter sovrascrivere i metodi quando necessario, avendo così la sicurezza di poter fare affidamento all'implementazione originale. Similmente a quanto avviene per il package `dataset`, alcuni attributi e metodi sono stati omessi dai diagrammi di classe per ottenere una maggiore chiarezza, benché vengano comunque analizzati nel seguito.

Il diagramma del package `problem` è visibile nella figura 6.6. Alcuni dei diagrammi rappresentano le classi concrete (o le interfacce) appartenenti a *jMetal* estese (o implementate) per ottenere l'integrazione dello stesso con *NewBestSub*. Nell'ambito del package, l'unico accesso diretto fra classi al suo interno avviene tra quelle che rappresentano problema e soluzione, poiché il primo ha bisogno di un riferimento alla soluzione corrente il quale viene sfruttato durante la fase di valutazione. Non sono necessarie ulteriori legami interni poiché il modello ha il compito di istanziare le quattro entità all'interno del metodo di risoluzione, mentre il framework gestisce internamente le interazioni fra le stesse, non appena il modello stesso avvia concretamente l'esecuzione dell'algoritmo scelto. C'è solo una piccola differenza rispetto a quanto descritto nella sezione 4.2. In tale sezione, infatti, viene indicata la classe `DefaultBinarySolution` come quella da estendere per rappresentare le soluzioni. In realtà, nello schema si vede come la classe `BestSubsetSolution` implementi un'interfaccia. Ciò avviene semplicemente perché `BestSubsetSolution` sovrascrive tutti i metodi di `DefaultBinarySolution`, per tale motivo è risultato conveniente eliminare il legame con tale classe ed implementare direttamente l'interfaccia utilizzata dalla classe concreta fornita dal framework.

La classe `BestSubsetSolution`, dunque, ha il compito di rappresentare le soluzioni analizzate nell'ambito della modellazione degli esperimenti *Best* e *Worst* come

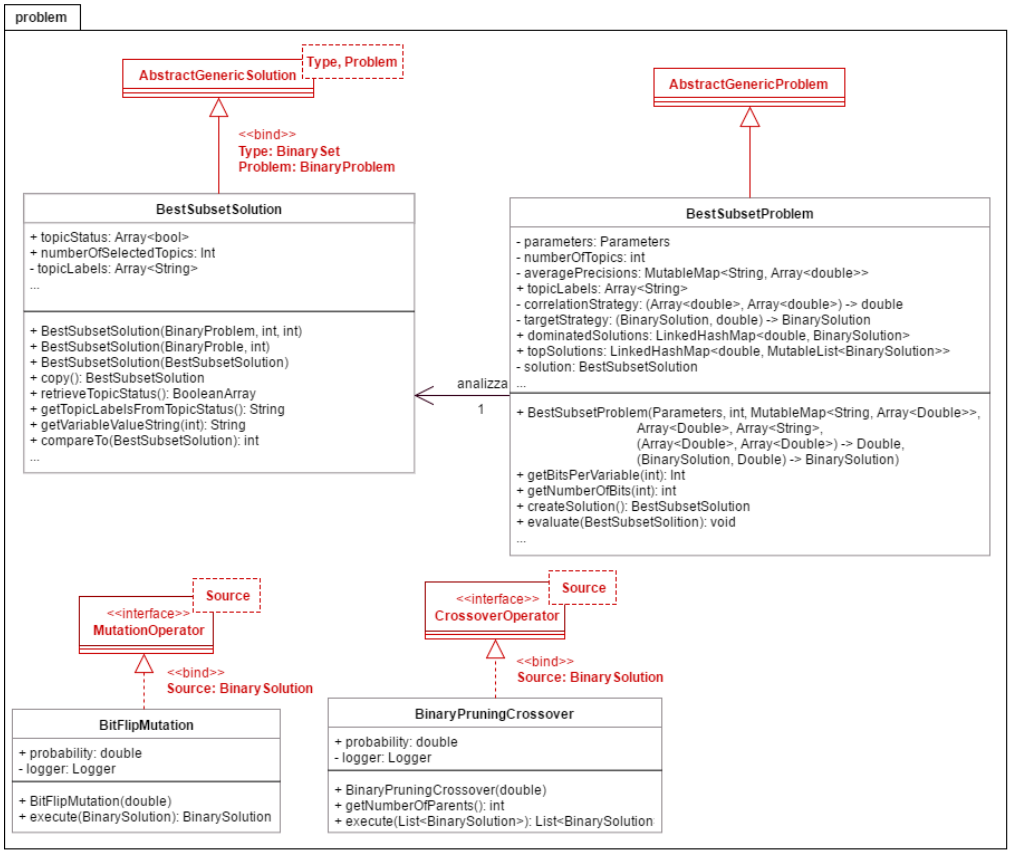


Figura 6.6: Diagramma delle classi contenute nel package `problem`.

problemi di ottimizzazione multi-obiettivo. Ogni soluzione, dunque, viene caratterizzata da un sottoinsieme di topic di una precisa cardinalità e tali sottoinsiemi vengono rappresentati come dei vettori binari. I tre attributi della classe, dunque, servono ad immagazzinare il vettore che descrive il sottoinsieme che caratterizza tale soluzione, la cardinalità del vettore stesso e le etichette dei topic selezionati. L'entità scelta per la rappresentazione tra quelle presenti nel diagramma visibile nella figura 4.2, dunque, è `BinarySolution`.

Per quanto riguarda i metodi, come prima cosa è necessario chiarire il ruolo dei tre costruttori indicati. Il primo di essi viene utilizzato quando è necessario che la soluzione generata abbia una precisa cardinalità e riceve tre parametri, i quali consistono in un riferimento al problema, nel numero di topic del dataset e nella cardinalità da generare. In particolare, tale costruttore viene sfruttato nella fase iniziale di generazione delle nuove soluzioni da parte del problema, per assicurare che ve ne sia almeno una per ciascuna cardinalità. Il secondo ha un comportamento

analogo al primo, dove l'unica differenza sta nel fatto che la cardinalità stessa viene determinata casualmente. Viene sfruttato quando ha il problema ha generato almeno una soluzione per ciascuna cardinalità e non deve più preoccuparsi di tale vincolo. In tal modo, il problema stesso può continuare a generarne di nuove fino a raggiungere la dimensione desiderata per la popolazione. Il terzo costruttore istanzia una nuova soluzione eseguendo una copia profonda di quella passata come parametro. Come scorciatoia può essere usato il metodo di copia, che invoca tale costruttore sulla soluzione stessa.

Vi è poi una serie di metodi d'utilità, sfruttati principalmente dai costruttori e dai metodi della classe che rappresenta il problema. Tali metodi servono a creare il vettore binario della soluzione corrente, ad impostare il valore di una delle sue componenti ed a restituirne un riferimento al chiamante. Ve ne sono altri due che vengono utilizzati per stampare le etichette dei topic selezionati dal dataset originale ed i valori delle componenti del vettore. Una terza categoria di metodi consiste nell'implementazione di quelli presenti nell'interfaccia `BinarySolution`, i quali permettono al problema di ottenere informazioni sulla struttura del vettore quali il numero di topic selezionati e la dimensione totale dello stesso; in altre parole, la cardinalità della soluzione corrente e quella massima presente nel dataset. Infine, vi sono altri tre metodi che vengono utilizzati per ordinare un insieme di soluzioni e verificare l'uguaglianza di quella corrente con quella passata come parametro.

La classe `BestSubsetProblem` rappresenta concretamente il problema multi-obiettivo mediante il quale vengono modellati gli esperimenti *Best* e *Worst*. L'interfaccia di riferimento implementata dalla classe concreta, tra quelle presenti nel diagramma visibile nella figura 4.3, è `BinaryProblem`, poiché il problema stesso gestisce solamente soluzioni caratterizzate da vettori binari.

Tra gli attributi di `BestSubsetProblem` si ritrovano diversi elementi già presenti nel modello. Ciò avviene perché essi vengono effettivamente utilizzati o prodotti all'interno della classe stessa. Nella prima categoria vi sono i parametri che il controller invia al modello e che quest'ultimo configura nell'istanza del problema. Essi, in particolare, vengono sfruttati per definire i metodi utilizzati per calcolare le correlazioni tra i valori di MAP originali e ridotti e per calcolare i valori delle funzioni obiettivo a seconda dell'esperimento scelto. Vi sono, inoltre, dei riferimenti alle strutture dati contenenti i valori originali di AP e quelli calcolati di MAP ed al numero totale di topic del dataset. Per quanto riguarda la seconda categoria, la classe traccia esplicitamente le migliori dieci soluzioni *dominate* individuate per ciascuna cardinalità. Una volta terminata la fase di valutazione, esse vengono for-

nite al modello. Vi sono poi altri due attributi, dove il primo viene utilizzato per mantenere un riferimento alla soluzione correntemente utilizzata, mentre il secondo indica la cardinalità da generate. Gli attributi rimanenti vengono utilizzati, infine, per scopi di logging.

Per quanto riguarda i metodi, all'interno del costruttore vengono solamente impostati nei relativi attributi gli elementi ricevuti dal modello, tra i quali figurano una copia dei parametri inviati dall'utente, i valori di *AP* e *MAP* ricavati dal dataset originale, le etichette dei topic ed i due metodi per calcolare i valori di correlazione e quelli delle funzioni obiettivo. Tutte queste informazioni, infatti, sono necessarie nella fase di valutazione delle soluzioni generate. Vi sono poi alcune segnature le quali indicano l'implementazione dell'interfaccia `BinaryProblem` ed il metodo che ha il compito di eseguire la fase di creazione dell'insieme di soluzioni da analizzare. Tale metodo viene invocato all'interno di jMetal per un numero di volte pari alla dimensione che la popolazione di individui deve raggiungere e ad ogni invocazione restituisce una nuova istanza di `BestSubsetSolution`. Esso genera esattamente una soluzione per cardinalità fino al raggiungimento del numero di topic del dataset originale e, quando ciò avviene, la cardinalità delle soluzioni generate da tale momento in poi viene determinata casualmente, fino al raggiungimento della dimensione desiderata per la popolazione di individui. Inoltre, non c'è alcun riferimento alla struttura dati utilizzata per immagazzinare le soluzioni durante le fasi di generazione e valutazione, in quanto tale informazione viene gestita internamente da *jMetal*.

L'ultimo metodo presente nel diagramma è quello con il compito di svolgere concretamente la fase di valutazione della soluzione ricevuta come parametro. Tale soluzione viene sottoposta alle azioni degli operatori di *Mutazione* e *Crossover* internamente al framework, e viene resa disponibile al metodo da esso. Nelle classi di *NewBestSub*, infatti, non c'è mai una chiamata esplicita ai metodi degli operatori e nemmeno al metodo di valutazione stesso, poiché sono tutte incapsulate nella classe che implementa l'algoritmo genetico *NSGA-II* . Il procedimento di valutazione di tale soluzione si può dividere in tre fasi. Nella prima, viene calcolata la correlazione tra la *MAP* del sottoinsieme di topic che la caratterizza e la *MAP* originale per la relativa cardinalità, impostando conseguentemente i valori delle funzioni obiettivo. Nella seconda viene creata una copia della soluzione e viene verificato se esiste già una soluzione *dominata* immagazzinata a parità di cardinalità. Se non esiste, la copia così creata assume tale ruolo, altrimenti viene confrontata con la soluzione già esistente per verificare quale, tra le due, è la migliore. Nella terza fase, un'ulteriore

copia della soluzione originale viene istanziata, per poi essere inserita nella lista delle migliori dieci soluzioni per la sua cardinalità. Non appena tale lista contiene undici elementi, essa viene ordinata per correlazione con un ordinamento decrescente, se l'esperimento è di tipo *Best*, o decrescente, se l'esperimento è di tipo *Worst*. A questo punto, l'ultimo elemento nella lista stessa viene eliminato. In tal modo, ad ogni iterazione essa contiene le dieci migliori soluzioni individuate fino a tale momento.

A questo è possibile descrivere gli operatori, cominciando con quello di *Mutazione*, implementato dalla classe `BitFlipMutation`. Nel costruttore viene passato come parametro un valore di probabilità, il quale viene semplicemente memorizzato nel rispettivo attributo. L'operatore, una volta configurato dal modello, viene utilizzato internamente dall'algoritmo per mutare la soluzione correntemente analizzata sfruttando l'apposito metodo, e tale soluzione è quella che viene inviata come parametro al metodo stesso. All'interno di esso, l'operatore genera un valore compreso tra zero ed uno e se tale valore è minore della probabilità ricevuta attraverso il costruttore esso muta concretamente la soluzione, mentre nel caso contrario la soluzione rimane intatta. La mutazione che viene eseguita dall'operatore è quella descritta nella sezione 3.5 ed il suo significato, in questo contesto, consiste nel selezionare un topic a caso appartenente al dataset originale ed includerlo o rimuoverlo dalla soluzione.

L'operatore di *Crossover* viene implementato da una classe la cui struttura simile a quella dell'operatore di *Mutazione*, ossia `BinaryPruningCrossover`. Le uniche differenze consistono nel fatto che è necessario un metodo aggiuntivo per poter comunicare al framework il numero di genitori utilizzato per creare le soluzioni figlie e che quello con il compito di eseguire concretamente il crossover riceve come parametro e restituisce una lista di soluzioni. Il procedimento che viene eseguito è quello descritto nella sezione 3.4. In questo contesto, il numero di genitori utilizzati è sempre pari a due e vengono generati altrettanti figli, il primo ponendo in AND le componenti dei genitori, il secondo ponendo tali componenti in OR.

Il significato dell'utilizzo di tali valori di probabilità è che l'azione degli operatori non dev'essere sempre possibile poiché può portare al verificarsi di effetti indesiderati, come descritto nelle sezioni 3.4 e 3.5.

6.4 Conclusioni

A questo punto la descrizione dei quattro package che costituiscono l'architettura di *NewBestSub* può dirsi completa. Ad ogni modo, può essere utile avere una visione d'insieme relativa a come gli elementi di tali package interagiscono tra di

essi, che vada oltre alle semplici dipendenze visualizzate nella figura 6.2. Una di queste interazioni da visualizzare, per esempio, è il modo in cui le classi del package `problem` vengono legate al modello, contenuto nel package `dataset`. Per tale motivo, nel diagramma visibile nella figura 6.7 è possibile osservare la struttura completa dell'architettura del software dove, per non comprometterne la leggibilità, gli elementi architetturali interni di *jMetal* estesi o implementati dalle classi del package `problem` sono stati rimossi.

Figura 6.7: Diagramma generale dell'architettura di *NewBestSub*.

Capitolo 7

Funzionalità di *NewBestSub*

Lo scopo di questo capitolo è descrivere le funzionalità fornite da *NewBestSub* fornisce, le limitazioni che lo affliggono e le tecnologie utilizzate durante la fase d'implementazione. In particolare, nella sezione 7.1 viene descritto come l'utente può interagire con esso in termini di processo analizzando i vari casi d'uso del sistema, la fase di deployment, il modo in cui è possibile eseguirlo concretamente (con quali opzioni da riga di comando) e le funzionalità aggiuntive disponibili. Nella sezione 7.2 vi è una discussione relativa all'esito della fase d'implementazione, nell'ambito della quale vi è un confronto con *BestSub*, l'analisi delle limitazioni che caratterizzano lo stesso *NewBestSub* e la presentazione delle tecnologie utilizzate durante le fase d'implementazione stessa e quella di progettazione architetturale.

7.1 Descrizione

Nel capitolo 6 *NewBestSub* viene analizzato in termini architetturali, descrivendo il ruolo delle varie componenti che lo caratterizzano, il modo in cui esse sono progettate ed implementate e le operazioni che possono svolgere. Per avere una panoramica completa del sistema, tuttavia, manca ancora una visione d'insieme relativa al modo in cui un utente può interagire con esso, a quali sono gli input che riceve e gli output che produce ed alla descrizione con un maggior livello di dettaglio del flusso di una singola esecuzione. Quello che si cerca di portare a termine in queste sezione, dunque, è un'analisi di *NewBestSub* a livello di *processo*.

7.1.1 Casi d'uso

Per facilitare un'analisi a livello di processo è utile visualizzare, in particolare, le interazioni che si verificano fra gli attori che partecipano ai processi stessi. Un modo efficace per farlo consiste nell'utilizzare un diagramma dei casi d'uso, come quello visibile nella figura 7.1. In tale diagramma viene indicato semplicemente un attore, che corrisponde ad un generico utente esterno che ha lo scopo di utilizzare *NewBestSub* per analizzare i propri dataset. I sottoprocessi collegati all'utente sono quelli che egli può effettivamente avviare i quali hanno, come sottoprocesso cardine, quello di risoluzione degli esperimenti. Tale sottoprocesso coinvolge quelli rimanenti, andando così a comporre l'intero flusso di esecuzione. Se la connessione ad un altro sottoprocesso presente nello schema viene etichettata con lo stereotipo `<<extends>>`, il significato è che quello da cui tale connessione si origina aggiunge dei passi opzionali al flusso d'esecuzione incapsulato all'interno di quello di destinazione. Nel seguito di tale analisi, inoltre, vengono approfondite le procedure svolte dal software accennate durante la fase di analisi dell'architettura del sistema nella sezione 6.3.

7.1.2 Deployment

A questo punto, prima di descrivere le opzioni da riga di comando mediante le quali un utente può personalizzare il comportamento di *NewBestSub*, è necessario illustrare i prerequisiti richiesti per l'esecuzione del software. *NewBestSub*, in

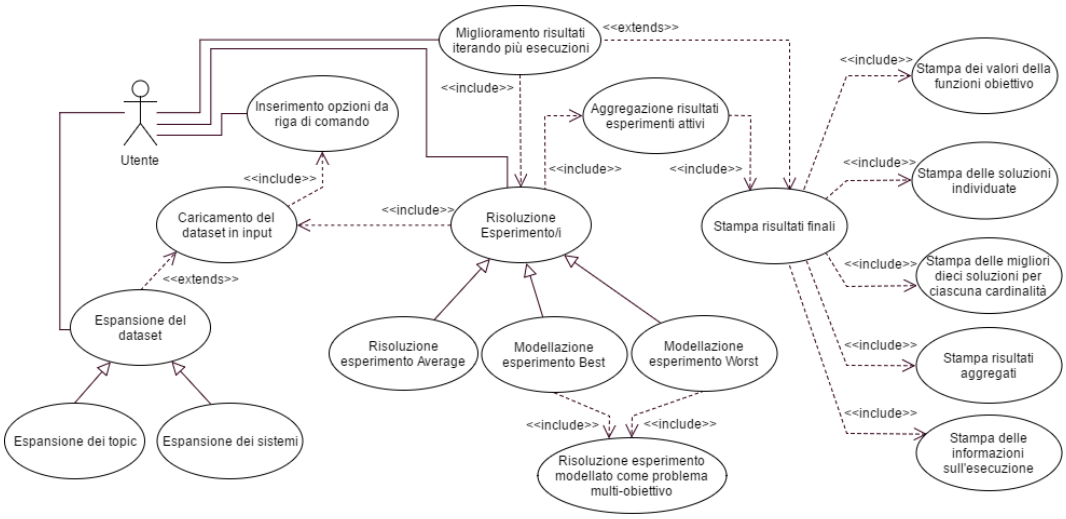


Figura 7.1: Diagramma dei casi d'uso che mostra le interazioni fra l'utente esterno e *NewBestSub*.

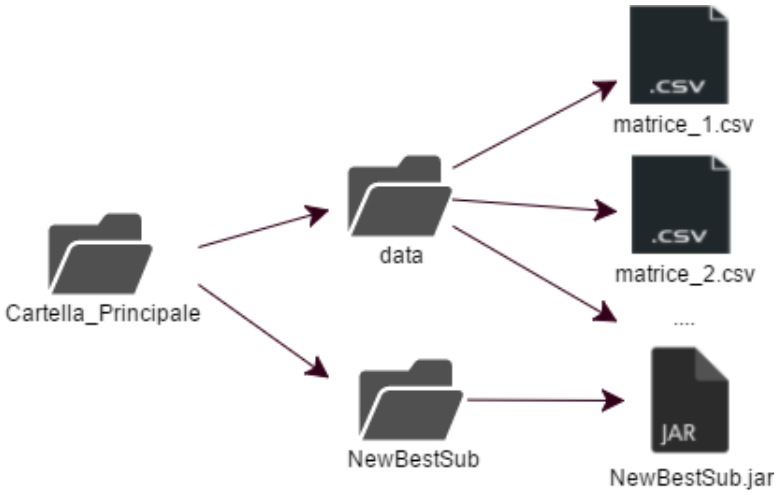


Figura 7.2: Organizzazione dei file richiesta sul file system per le esecuzioni di *NewBestSub*.

particolare, viene distribuito all'interno di un archivio *JAR*, il quale include tutte le dipendenze necessarie per il suo funzionamento. Per poterlo utilizzare correttamente, l'unico vincolo richiesto è che la sua collocazione sul file system rispetti la struttura rappresentata nella figura 7.2.

La cartella principale che funge da contenitore, in particolare, può avere un nome qualsiasi ma è necessario che contenga una cartella chiamata `data`, poiché il software cerca i dataset in input al suo interno. Inoltre, ve ne deve essere una seconda chiamata obbligatoriamente `NewBestSub`, la quale identifica il punto in cui si trova l'archivio *JAR* contenente le classi compilate di *NewBestSub*. L'archivio stesso, al contrario, può essere rinominato a piacimento, proprio perché la sua posizione viene determinata dalla cartella precedentemente citata. I dataset richiesti in input dal software sono delle matrici sistemi per topic dove ciascuna cella contiene i valori della metrica per l'efficacia dei sistemi di IR utilizzata dal dataset correntemente analizzato ed il modo più naturale per rappresentare tali matrici consiste nell'uso del formato *CSV*. Per tale motivo, la cartella `data` contiene tanti file con estensione `.csv` tante quante sono le matrici da analizzare. Ognuna di esse, dunque, rappresenta un preciso dataset e costituisce l'input di una singola esecuzione del software. Inoltre, un frammento di una di tali matrici può essere osservato nel listato 7.1.

Una volta terminata l'esecuzione di *NewBestSub* su un singolo dataset, vengono prodotte ulteriori cartelle contenenti file aggiuntivi i quali, a loro volta, contengono tutti i risultati ottenuti nell'ambito dell'esecuzione corrente. Al termine della sezione

viene presentata una figura analoga alla 7.2 in grado di descrivere il modo in cui i risultati stessi vengono posizionati sul file system. Tale figura viene presentata al termine della sezione poiché prima è necessario descrivere i sottoprocessi che portano all'ottenimento di tali risultati.

```
APS426,446,404,431,406,437,448,449,434,438,...
INQ604,0.08,0.14,0.18,0.16,0.40,0.25,0.60,0.11,0.35,0.22,...
ibmg99a,0.26,0.74,0.12,0.10,0.44,0.46,0.42,0.69,0.34,0.19,...
Mer8Adtd4,0.50,0.25,0.19,0.30,0.41,0.90,0.10,0.97,0.28,0.31,...
```

Listato 7.1: Frammento di dataset in input per una singola esecuzione di *NewBestSub*

7.1.3 Opzioni da Riga di Comando

Le opzioni da riga di comando che un utente può utilizzare per personalizzare il comportamento di *NewBestSub* sono visibili nella tabella 7.1. In particolare, per ogni opzione vengono riportate le forme brevi ed estese, una descrizione di cosa permettono di ottenere ed i valori consentiti. In alcuni casi, essi sono puntuali e vanno scelti fra un numero finito di alternative, mentre in altri è possibile utilizzare intere categorie di valori. Per un esempio delle due casistiche, si vedano le opzioni `--corr` e `--iter`; i valori possibili per la prima sono "Pearson" e "Kendall" mentre per la seconda consistono in un numero intero e positivo qualsiasi. Vi sono, inoltre, due ulteriori colonne le quali indicano, rispettivamente, l'obbligatorietà o meno di ciascuna opzione ed eventuali dipendenze con quelle rimanenti. Nell'ultima colonna della tabella, infine, vengono riportati casi particolari nell'uso di una determinata opzione poiché, ad esempio, il suo utilizzo può rendere obbligatorio quello di alcune tra quelle rimanenti.

Il ruolo di ciascuna opzione da riga di comando viene dettagliato nella descrizione del sottoprocesso in cui essa viene effettivamente sfruttata, tuttavia è possibile svolgere alcune precisazioni. In particolare, il valore dell'opzione `--fi` rappresenta l'unico path che l'utente deve inserire manualmente poiché risulta necessario per selezionare il dataset in input per l'esecuzione corrente. Tale path, inoltre, è relativo alla cartella `data` e l'estensione del file contenente tale dataset non viene richiesta. Facendo riferimento alla figura 7.2, ad esempio, per indicare in input il dataset `matrice_1.csv` è sufficiente assegnare all'opzione `--fi` il valore `"matrice_1"`. Infine, osservando il diagramma visibile nella figura 6.5, è possibile capire quali opzioni rappresentano i parametri necessari all'esperimento incapsulati all'interno del mo-

dello e quali vengono utilizzate all'esterno di esso. L'uso delle opzioni `--es` o `--et` accoppiate all'opzione `--mx` o l'uso dell'opzione `--mr` permette di eseguire le funzionalità aggiuntive menzionate nella sezione 6.3. Tali funzionalità vengono analizzare dopo aver illustrato i sottoprocessi coinvolti in un'esecuzione standard del sistema poiché consistono nell'iterare più volte tali processi dove, a seconda dei casi, viene pre-elaborato il dataset in input durante ciascuna iterazione oppure vengono fusi assieme i risultati ottenuti da ciascuna di tali iterazioni, il cui numero viene indicato dall'utente.

Tabella 7.1: Opzioni da riga di comando utilizzabili per l'esecuzione di *NewBestSub*.

Breve	**Estesa**	**Descrizione**	**Valori**	**Obb.**	**Dip.**	**Note**
`--fi`	`--fileIn`	Path relativo per il file del dataset a partire dalla cartella `data`. *Non va specificata* l'estensione di tale file.	Stringa rappresentante un path.	Si	/	/
`--c`	`--corr`	Metodo da utilizzare nel calcolo della correlazione.	Pearson, Kendall	Si	/	/
`--t`	`--targ`	Esperimento da eseguire.	Best, Worst, Average, All	Si	/	/
`--l`	`--log`	Livello della frequenza di logging.	Verbose, Limited, Off	Si	/	/
`--i`	`--iter`	Numero di iterazioni con cui *jMetal* valuta le soluzioni generate.	Numero intero positivo.	No	`--t`	Obbligatoria se `--t` ha valore: Best, Worst, All.

Continua nella pagina seguente

Tabella 7.1: Opzioni da riga di comando utilizzabili per l'esecuzione di *NewBestSub*. (Cont.)

Breve	**Estesa**	**Descrizione**	**Valori**	**Obb.**	**Dip.**	**Note**
`--r`	`--rep`	Numero di ripetizioni con cui calcolare una singola cardinalità.	Numero intero positivo.	No	`--t` `--mx`	Obbligatoria se `--t` ha valore: Average, All.
`--po`	`--pop`	Dimensione della popolazione iniziale da generare.	Numero intero positivo.	No	`--t`	Obbligatoria se `--t` ha valore: Best, Worst, All. Il suo valore dev'essere $\geq$ di quello di `--mx`, se usata. Il suo valore dev'essere maggiore del numero di topic del dataset.
`--pe`	`--perc`	Intervallo di percentili da calcolare.	Due numeri interi positivi separati da virgola. Il secondo $\geq$ del primo. Es: 25,75	No	`--t`	Obbligatoria se `--t` ha valore: Average, All.
`--et`	`--expt`	Numero di topic finti da aggiungere ad ogni espansione.	Numero intero positivo.	No	`--mx`	Rende obbligatorio l'uso di `--mx`. Il suo valore dev'essere $\leq$ di quello di `--mx`.

Continua nella pagina seguente

Tabella 7.1: Opzioni da riga di comando utilizzabili per l'esecuzione di *NewBestSub*. (Cont.)

Breve	Estesa	Descrizione	Valori	Obb.	Dip.	Note
`--es`	`--exps`	Numero di sistemi finti da aggiungere ad ogni espansione.	Numero intero positivo.	No	`--mx`	Rende obbligatorio l'uso di `--mx`. Il suo valore dev'essere $\leq$ di quello di `--mx`.
`--mx`	`--max`	Numero massimo di topic (sistemi) da raggiungere durante la fase di espansione.	Numero intero positivo.	No	`--et` `--es` `--po`	Resa obbligatoria dall'uso di `--et` o `--es`. Il suo valore dev'essere $\leq$ di quello di `--po`
`--mr`	`--mrg`	Numero di esecuzioni del programma da lanciare e di cui fondere i risultati.	Numero intero positivo.	No	/	/

7.1.4 Esecuzione di *NewBestSub*

Prima di proseguire con la descrizione dei sottoprocessi non ancora analizzati tra quelli presenti nella figura 7.1, è necessario chiarire il modo in cui è possibile eseguire concretamente *NewBestSub*. Come già descritto, esso viene distribuito in un archivio *JAR*, perciò per poterlo eseguire è necessario avere una *Java Virtual Machine* installata sul proprio calcolatore. Il comando da utilizzare da terminale per lanciare il programma stesso, dunque, assume il seguente formato:

```
java -jar [Path per il file .jar] [opzioni da riga di comando]
```

Nel seguito, è possibile osservare alcuni esempi di configurazioni delle opzioni da riga di comando le quali portano ad ottenere diverse esecuzioni del programma stesso. In particolare, vengono rappresentati cinque diversi casi, ovvero:

1. esecuzione di un esperimento di tipo *Best* sul dataset `matrice_1.csv`;

2. esecuzione di un esperimento di tipo *Average* sul dataset `matrice_1.csv`;
3. esecuzione di tutti gli esperimenti sul dataset `matrice_1.csv`;
4. esecuzione di tutti gli esperimenti sul dataset `matrice_1.csv`, ripetuti cinque volte per migliorare i risultati ottenuti;
5. esecuzione di tutti gli esperimenti sul dataset `matrice_1.csv`, ripetuti aggiungendo ad ogni iterazione venti topic finti, fino ad un massimo di ottocento topic.

I casi dell'elenco precedente si traducono nelle seguenti configurazioni di opzioni da riga di comando:

1. `-fi "matrice_1" -c "Pearson" -po 2000 -i 100000 -t "Best" -l Limited`
2. `-fi "matrice_1" -c "Pearson" -r 2000 -t "Average" -pe 1,100 -l Limited`
3. `-fi "matrice_1" -c "Pearson" -po 2000 -r 2000 -i 100000 -t "All" -pe 1,100 -l Limited`
4. `-fi "matrice_1" -c "Pearson" -po 2000 -i 100000 -r 2000 -t "All" -pe 1,100 -l Limited -mr 5`
5. `-fi "matrice_1" -c "Pearson" -po 2000 -i 100000 -r 2000 -t "All" -pe 1,100 -l Limited -et 20 -mx 800`

Assumendo di non aver alterato la struttura della versione di *NewBestSub* ottenuta tramite download dalla repository e di rispettare la struttura sul file system rappresentata nella figura 7.2, il comando completo per eseguire il programma nel terzo dei casi precedentemente descritti, ad esempio, è quello indicato nel listato 7.2 e va eseguito solamente una volta posizionatisi nella cartella contenente l'archivio *JAR*. I casi rimanenti vengono lasciati al lettore.

```
java -jar NewBestSub-1.0-jar-with-dependencies.jar -fi "matrice_1" -c "
    Pearson" -po 2000 -r 2000 -i 100000 -t "All" -pe 1,100 -l Limited
```

Listato 7.2: Comando da terminale per lanciare un'esecuzione standard di *NewBestSub*.

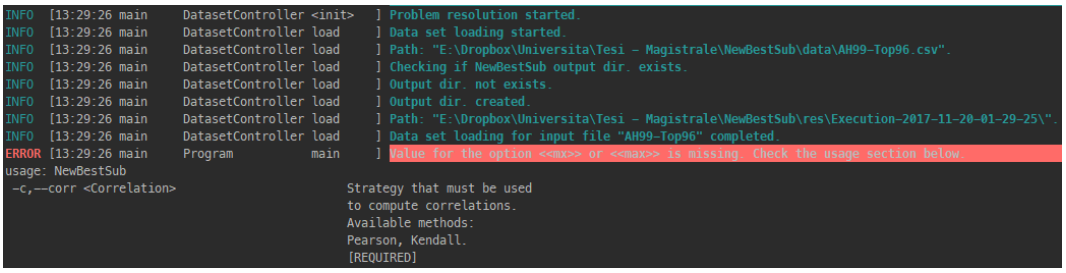


Figura 7.3: Tipici messaggi di log stampati su terminale nel corso di un'esecuzione di *NewBestSub* con opzioni da riga di comando mancanti o con valori errati.

Il sottoprocesso successivo all'inserimento delle opzioni da riga di comando da parte dell'utente consiste nell'operazione di parsing delle stesse. In questa fase, dunque, viene semplicemente estratto e letto il valore di ciascuna di tali opzioni e, inoltre, vengono catturate eventuali eccezioni, qualora i valori inseriti non rispettassero i vincoli della tabella 7.1. Per osservare un tipico messaggio d'errore dovuto a tali violazioni si supponga, ad esempio, di voler espandere il numero dei topic del dataset in input, chiamando in causa l'opzione `--et`, senza inserire, però, il numero massimo raggiungibile da essi mediante l'opzione `--mx`. Nella figura 7.3 è possibile il relativo messaggio d'errore. In particolare, vi sono alcuni messaggi di log per operazioni andate a buon fine, seguiti da quello d'errore con la relativa descrizione ed un frammento della sezione d'aiuto che viene stampata ad ogni eccezione catturata.

I due sottoprocessi di gestione delle opzioni da riga di comando sono un prerequisito per il caricamento del dataset in input. In questa fase, in particolare, viene istanziato un modello per ciascun esperimento scelto poiché, come si vede osservando i valori assegnabili all'opzione `--t` indicati nella tabella 7.1, è possibile richiedere a *NewBestSub* di svolgerli tutti e tre in un'unica esecuzione. Una volta fatto ciò, ciascun modello attivo analizza il dataset così caricato per svolgere l'esperimento assegnato. Tale fase di analisi consiste nella scansione del file `.csv` indicato dall'utente riga per riga, dove ciascuna di esse contiene un vettore di AP, una serie etichette che identificano i topic a cui il dataset stesso fa riferimento ed una singola etichetta che rappresenta il nome di un sistema. Ogni riga di tale file `.csv` rappresenta, dunque, tutte le informazioni relative al risultato del testing di un preciso sistema di IR sui topic del dataset. Una volta archiviate tali informazioni all'interno delle apposite strutture dati, vengono calcolati i valori di MAP per ciascuno dei sistemi di IR presenti nel dataset stesso. Tali valori, successivamente, vengono immagazzinati in un vettore apposito il quale ha, dunque, una dimensione pari al numero di sistemi

presenti nel dataset.

I sottoprocessi descritti fino a questo punto sono dei prerequisiti per quello di risoluzione degli esperimenti scelti. All'interno di tale sottoprocesso l'esecuzione del sistema può prendere, per ciascun modello, tre percorsi alternativi, come si può vedere nel diagramma visibile nella figura 7.1. Il valore discriminante nella scelta di uno dei due percorsi è il tipo di esperimento assegnato al modello corrente. Naturalmente, se si è scelto di eseguire tutti gli esperimenti, prima o poi il sistema svolge le operazioni descritte da ogni percorso rappresentato nel diagramma.

In particolare, se l'esperimento è di tipo *Average* esso viene svolto direttamente dal modello, senza fare affidamento alle funzionalità di *jMetal*, poiché si tratta di un problema facile. Il procedimento che viene svolto consiste nel generare, per ciascuna cardinalità, una singola soluzione, il cui sottoinsieme di topic ha dimensione pari alla cardinalità corrente, dove tali topic vengono selezionati casualmente. Successivamente, viene calcolato il valore di correlazione, secondo il metodo scelto dall'utente[1], fra il vettore ottenuto calcolando le *MAP* per ciascun sistema utilizzando solamente i topic selezionati dalla soluzione stessa ed il vettore originale. Questo procedimento viene iterato per un numero di ripetizioni pari a quello indicato dall'utente[2], archiviando ciascun valore di correlazione così calcolato in un vettore apposito, di dimensione pari al numero di ripetizioni stesso. Il valore finale di correlazione viene ottenuto sommando quelli archiviati nel vettore intermedio e dividendo tale somma per la dimensione di quest'ultimo. Tale procedimento viene ripetuto per un numero arbitrario di volte perché consente alla media di convergere verso un dato valore permettendo, in tal modo, di ottenere una distribuzione più stabile di valori di correlazione per ciascuna cardinalità. Prima di passare alla cardinalità successiva, il vettore intermedio viene nuovamente analizzato poiché all'interno di esso viene calcolata la posizione di ciascun percentile appartenente all'intervallo specificato dall'utente[3].

L'utente può specificare come opzione da riga di comando il numero di ripetizioni poiché esso costituisce un collo di bottiglia nell'esecuzione dell'esperimento, in quanto un valore troppo alto aumenta drasticamente il tempo d'esecuzione, mentre un valore troppo basso porta ad avere valori di correlazione poco stabili. Il tutto, inoltre, viene influenzato dalla capacità computazionale del calcolatore sul quale viene svolto l'esperimento. L'utente, dunque, dev'essere in grado di indicare tale

[1] Il valore dell'opzione `--c`.

[2] Il valore dell'opzione `--r`.

[3] Il valore dell'opzione `--pe`.

valore al fine di cercare il compromesso giusto fra stabilità dei risultati e tempo d'esecuzione a seconda delle proprie disponibilità hardware. Una volta terminata la fase di generazione e valutazione delle soluzioni esse vengono considerate come soluzioni *non dominate*, poiché ve ne sarà solamente una per ciascuna cardinalità, e semplicemente copiate nella struttura dati contenente quelle finali dell'esperimento, come descritto nella sezione 6.3.

Se l'esperimento è di tipo *Best* o *Worst*, il sottoprocesso viene scomposto in una fase di modellazione ed in una fase di risoluzione vera e propria, come si può vedere nel diagramma presente nella figura 7.1, poiché non è possibile risolvere il problema direttamente ed è necessario modellarlo come problema di ottimizzazione multi-obiettivo affidandosi a *jMetal*.

La prima fase consiste nell'effettiva modellazione dell'esperimento come problema di ottimizzazione multi-obiettivo ed avviene istanziando gli elementi architetturali del framework stesso con i parametri richiesti e configurando l'algoritmo genetico scelto per la risoluzione del problema stesso con tali istanze. In altre parole, si fornisce a *NSGA-II* gli operatori che deve utilizzare per manipolare le soluzioni nel corso dell'esecuzione. Concretamente, viene creata un'istanza della rappresentazione del problema, alla quale vengono inviati i parametri inseriti dall'utente, la matrice di *AP*, il vettore di *MAP*, le etichette dei topic, il metodo per calcolare i valori di correlazione e quello per calcolare i valori delle funzioni obiettivo. Successivamente, vengono istanziati gli operatori di *Crossover* e *Mutazione*, i quali ricevono come parametro un valore di probabilità fisso pari a, rispettivamente, 0.7 e 0.3, secondo i principi descritti nelle sezioni 3.4 e 3.5. Inoltre, viene istanziato anche l'operatore di *Selezione*, il quale non richiede alcun parametro esterno per poter svolgere le proprie operazioni. Infine, vengono impostati gli ultimi parametri di cui l'algoritmo ha bisogno, i quali consistono nella dimensione della popolazione da generare e nel numero massimo di iterazioni che possono essere svolte dall'algoritmo stesso[4].

La seconda fase comporta l'effettiva risoluzione dell'esperimento e viene svolta avviando l'algoritmo configurato con gli operatori precedentemente istanziati ed i parametri ricevuti dall'utente. L'esecuzione di tale algoritmo si articola nel flusso già descritto nella sezione 6.3, ossia in una fase iniziale di generazione delle soluzioni seguita da quella di valutazione delle soluzioni stesse, da svolgersi una volta terminate le manipolazioni effettuate dagli operatori. Una volta portato a termine tale flusso, l'algoritmo restituisce la lista di tutte le soluzioni *non dominate* ed un insieme di informazioni legate all'esecuzione concreta quali tempo di completamen-

[4] I valori delle opzioni `--po` e `--i`.

to e nome del thread utilizzato, mentre l'istanza del problema tiene traccia della lista delle soluzioni *dominate* e delle migliori dieci soluzioni ottenute per ciascuna cardinalità. Successivamente, le liste di soluzioni *non dominate* e *dominate* vengono fuse in un'unica struttura e, come ultimo passo, viene calcolata la distribuzione dei topic all'interno delle soluzioni rimaste dopo l'operazione di fusione. Tali soluzioni rappresentano, dunque, le migliori configurazioni di topic individuate nell'ambito dell'esecuzione corrente in grado di soddisfare l'obiettivo dell'esperimento assegnato al modello di appartenenza, per ciascuna cardinalità. Le operazioni di fusione e di calcolo della distribuzione dei topic, in particolare, sono quelle descritte nella sottosezione 6.3.3.

L'ultima operazione ad essere svolta, indipendentemente dall'esperimento scelto, consiste nell'ordinamento delle soluzioni finali in base alla loro cardinalità ed il risultato che viene restituito dalla fase di risoluzione dell'esperimento stesso consiste in una tripla composta dall'esito dell'operazione di ordinamento sulle soluzioni finali, dalle migliori dieci soluzioni individuate per ciascuna cardinalità (già ordinate poiché memorizzate in un dizionario) e da un ulteriore tripla contenente informazioni legate all'esecuzione concreta. In particolare, tali informazioni consistono nel tipo di esperimento portato a termine, nel nome del thread utilizzato ed nel tempo impiegato per portare a termine la computazione.

Il sottoprocesso successivo tra quelli presenti nel diagramma visibile nella figura 7.1 consiste nell'aggregazione dei risultati degli esperimenti scelti e precede la stampa di quelli disponibili all'interno di ciascun modello. L'operazione di aggregazione viene svolta per produrre un'unica visuale sui risultati stessi e consiste nell'effettuare una scansione cardinalità per cardinalità, contemporaneamente, di ciascuna struttura dati dei modelli attivi contenente risultati di una delle diverse tipologie dove, durante la scansione stessa, vengono svolte operazioni specifiche per la tipologia correntemente analizzata di risultato. Tali strutture dati sono quelle contenenti le soluzioni finali ordinate per cardinalità, le distribuzioni dei topic contenuti all'interno di esse[5] e l'elenco dei percentili, anch'esso ottenuto a partire dalle soluzioni finali[6]. Per quanto riguarda i valori di correlazione, le operazioni svolte durante la scansione consistono semplicemente nel copiare all'interno della struttura dati dedicata a tale scopo quelli degli esperimenti attivi, mentre per quanto riguarda la distribuzione dei topic ciò che viene svolto consiste nel copiare all'interno della struttura dati dedicata a tale scopo quella presente all'interno del modello utilizzato per

[5]Se gli esperimenti attivi sono di tipo *Best* o *Worst*.

[6]Se tra gli esperimenti attivi figura quello di tipo *Average*.

svolgere l'esperimento *Best* o *Worst*, qualora uno solo tra i due risulti attivo. Se, al contrario, risultano attivi entrambi i modelli, le due distribuzioni originali vengono fuse. Un topic, dunque, può ottenere l'etichetta `BW` se è presente nelle soluzioni finali della cardinalità correntemente analizzata presenti nelle strutture dati di entrambi i modelli. L'elenco dei percentili, infine, è una semplice copia di quello contenuto nel modello utilizzato per svolgere l'esperimento *Average*.

Una volta completato il sottoprocesso di aggregazione, i modelli attivi vengono scansionati nuovamente per aggregare, questa volta, le informazioni legate all'esecuzione concreta degli esperimenti. Tale operazione consiste, semplicemente, nell'archiviare in una nuova ed unica struttura dati una copia dei parametri dell'esecuzione corrente e del tempo impiegato per risolvere il problema, per ciascun esperimento scelto. Successivamente, non vengono prodotti ulteriori dati ed è possibile procedere con il sottoprocesso di stampa dei risultati finali.

La fase di stampa dei risultati finali, come si vede nel diagramma visibile nella figura 7.1, si compone di quattro ulteriori sottoprocessi, che si differenziano semplicemente per il tipo di risultato stampato. Concretamente, l'attività di stampa consiste nello scrivere i dati contenuti nella struttura dati fornita di volta in volta in un file `.csv`. Come già descritto precedentemente, alcune tipologie di risultato sono proprie di ciascun modello, mentre altre prendono in considerazione globalmente tutti quelli attivi nell'esecuzione corrente. I risultati che vengono prodotti, dunque, sono:

1. i valori delle funzioni obiettivo per ciascuna soluzione finale;
2. la composizione del sottoinsieme di topic che caratterizza ciascuna soluzione finale;
3. le dieci migliori soluzioni per ogni cardinalità (solo per gli esperimenti *Best* e *Worst*);
4. il risultato dell'aggregazione dei risultati per tutti gli esperimenti scelti;
5. il risultato dell'aggregazione delle informazioni legate all'esecuzione concreta per tutti gli esperimenti scelti.

Nel caso delle informazioni legate all'esecuzione concreta, nel rispettivo file viene scritta una riga per ciascun esperimento scelto mentre in quelli contenenti le migliori dieci soluzioni per ogni cardinalità vi sono tante righe quante sono le soluzioni finali effettivamente trovate per ciascuna di tali cardinalità. Nei file rimanenti, infine, vi

è esattamente una riga per ciascuna cardinalità e gli elementi contenuti in una data riga variano a seconda dei casi. I file prodotti durante le fasi di stampa facenti riferimento alle tipologie dei risultati indicate nell'elenco precedente vengono denominati con le seguenti convenzioni:

1. `nomeDataset-metodoCorrelazione-numeroTopic-numeroSistemi-parametriEsperimento-tipoEsperimento-Fun.csv`

2. `nomeDataset-metodoCorrelazione-numeroTopic-numeroSistemi-parametriEsperimento-tipoEsperimento-Var.csv`

3. `nomeDataset-metodoCorrelazione-numeroTopic-numeroSistemi-parametriEsperimento-tipoEsperimento-Top-10-Solutions.csv`

4. `nomeDataset-metodoCorrelazione-numeroTopic-numeroSistemi-parametriEsperimentiAttivi-All-Final.csv`

5. `nomeDataset-metodoCorrelazione-numeroTopic-numeroSistemi-parametriEsperimentiAttivi-All-Info.csv`

L'elemento "parametriEsperimento" viene espanso secondo una convenzione nella forma "numeroIterazione-dimensionePopolazione" per gli esperimenti *Best* e *Worst*, mentre per l'esperimento *Average* essa diventa "numeroRipetizioni". Si supponga, a questo punto, di eseguire il programma con la configurazione di opzioni da riga di comando descritta nel listato 7.2. I file prodotti da tale esecuzione, la quale richiede di svolgere tutti gli esperimenti, sono i seguenti:

- `Matrice1-Pearson-50-96-2000-Average-Fun.csv`
- `Matrice1-Pearson-50-96-2000-Average-Var.csv`
- `Matrice1-Pearson-50-96-100000-2000-Best-Fun.csv`
- `Matrice1-Pearson-50-96-100000-2000-Best-Var.csv`
- `Matrice1-Pearson-50-96-100000-2000-Best-Top-10-Solutions.csv`
- `Matrice1-Pearson-50-96-100000-2000-Worst-Fun.csv`

- `Matrice1-Pearson-50-96-100000-2000-Worst-Fun.csv`
- `Matrice1-Pearson-50-96-100000-2000-Worst-Top-10-Solutions.csv`
- `Matrice1-Pearson-50-96-100000-2000-2000-All-Final.csv`
- `Matrice1-Pearson-50-96-100000-2000-2000-All-Info.csv`
- `Execution.log`

Al termine di ciascuna esecuzione di *NewBestSub*, i file prodotti vengono immagazzinati all'interno di una cartella chiamata `Execution` alla quale viene aggiunto, come suffisso, il timestamp in cui l'esecuzione stessa è stata avviata, per evitare la sovrascrittura dei risultati ottenuti da quelle precedenti. Ognuna di tali cartelle viene creata all'interno di un'ulteriore cartella chiamata `res`. Qualora essa non esistesse, viene creata autonomamente dal programma. Infine, viene creato un ultimo file il quale contiene tutti i messaggi di log e viene salvato in una seconda cartella chiamata `Execution` dove, anche in questo caso, il timestamp in cui l'esecuzione è stata avviata viene aggiunto come suffisso. Tale cartella, inoltre, viene contenuta a sua volta all'interno di un'ulteriore cartella chiamata `log`. Al termine dell'esecuzione precedentemente riportata, la situazione sul file system descritta nella figura 7.2 si trasforma in quella descritta nella figura 7.4.

Una volta completate le operazioni descritte fino a questo punto, il flusso d'esecuzione standard di *NewBestSub* può considerarsi completo. Nel diagramma visibile nella figura 7.1, tuttavia, sono presenti alcuni sottoprocessi mai analizzati nel corso dei paragrafi precedenti. Tali sottoprocessi, infatti, sono quelli legati alle funzionalità aggiuntive già illustrate nella sezione 6.3. Come precisato durante la descrizione delle opzioni da riga di comando, esse consistono nell'iterazione di più esecuzioni standard del software dove, per ciascuna di tali iterazioni, viene pre-elaborato il dataset in input oppure vengono fusi i risultati ottenuti.

7.1.5 Funzionalità Aggiuntive

Le funzionalità che si affidano ad una pre-elaborazione del dataset sono quelle che consentono di espandere il dataset stesso, come descritto nel diagramma visibile nella figura 7.1, con topic o sistemi generati casualmente. Esse sono state sviluppare per poter testare l'efficienza del sistema dal punto di vista del tempo d'esecuzione, a fronte di un aumento lungo una delle due dimensioni dei dati di input. Tali

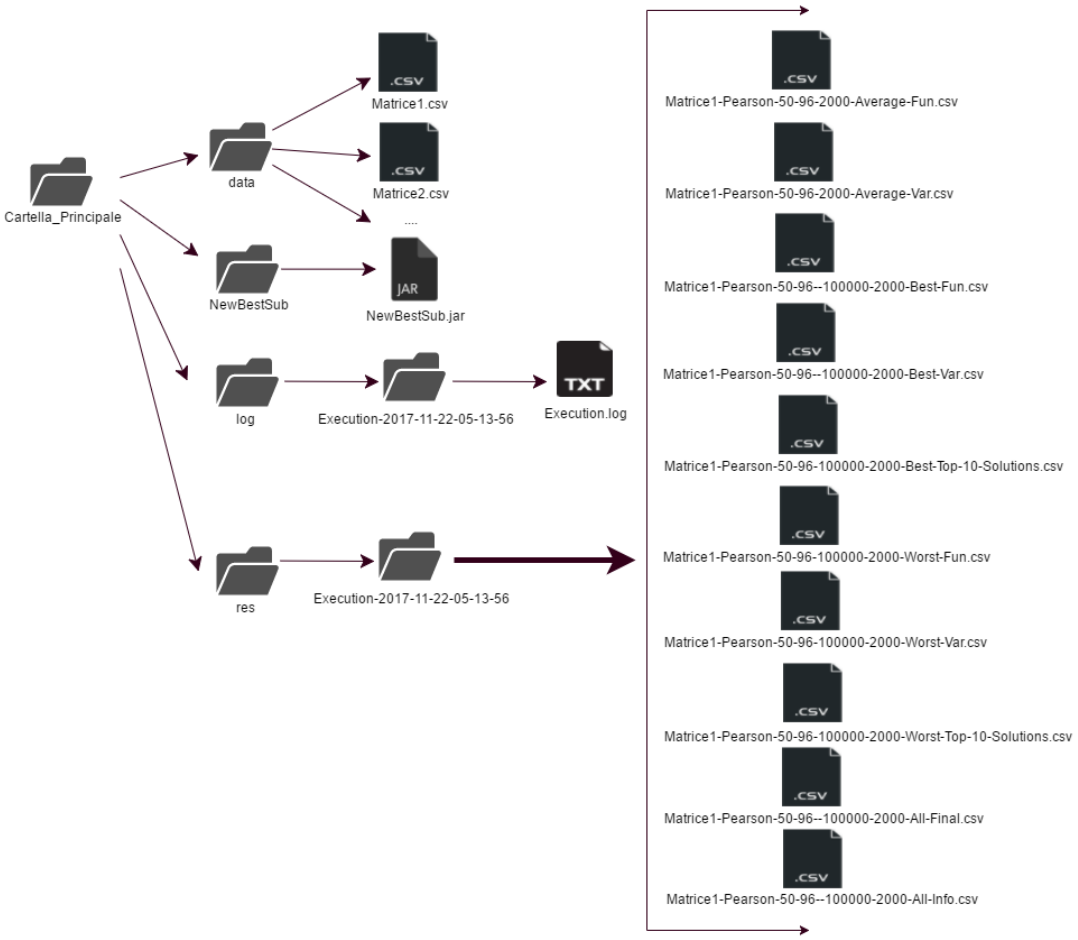


Figura 7.4: Organizzazione sul file system dei file contenenti i risultati di un'esecuzione standard di *NewBestSub*.

dimensioni corrispondono al numero di topic ed al numero di sistemi da valutare ed il principio utilizzato per incrementare il primo di essi consiste nell'aggiungere ad ogni sistema esistente, rispettivamente, tanti valori di AP quanti sono i topic che si desidera aggiungere e, per quanto riguarda il secondo, tanti nuovi vettori di AP quanti sono i sistemi che si desidera aggiungere. Poiché si tratta di un test legato all'efficienza, non è necessario avere dati reali, per cui i nuovi elementi vengono generati casualmente, secondo la procedura descritta nella sottosezione 6.3.3.

Per realizzare tutto ciò, l'utente ha la possibilità di indicare, mediante le opzioni da riga di comando, un coefficiente d'espansione del numero di topic (sistemi) ed

il numero massimo di essi oltre il quale non è possibile continuare l'espansione[7]. I due relativi sottoprocessi, dunque, consistono nell'iterare l'esecuzione standard del sistema più volte dove, prima di ciascuna iterazione, il dataset viene espanso per un numero di topic (sistemi) pari al coefficiente indicato mediante una procedura di generazione casuale degli stessi fino a raggiungere tale numero massimo. Ad ogni esecuzione, dunque, *NewBestSub* stesso genera i risultati descritti nella sottosezione 7.1.4, resi univoci per ciascuna di esse dalla convenzione utilizzata per denominare gli stessi, poiché il numero di topic (sistemi) varia di iterazione in iterazione.

Si supponga, a questo punto, di eseguire il programma con la configurazione di opzioni da riga di comando descritta dal listato 7.2 dove, in aggiunta, il dataset in input viene espanso di venti topic alla volta fino ad un massimo di cinquecento, ottenendo quella descritta nel listato 7.3. Una volta terminate tutte le esecuzioni, la situazione sul file system rappresentata nella figura 7.2 diventa quella rappresentata nella figura 7.5.

```
java -jar NewBestSub-1.0-jar-with-dependencies.jar -fi "matrice_1" -c "
    Pearson" -po 2000 -r 2000 -i 100000 -t "All" -pe 1,100 -l Limited -et 20
     -mx 500
```

Listato 7.3: Comando da terminale per lanciare un'esecuzione di *NewBestSub* espandendo i topic del dataset in input di volta in volta.

L'ultima funzionalità del software a non risultare ancora descritta è quella relativa al miglioramento dei risultati, come si può notare osservando diagramma visibile nella figura 7.1. Il principio di fondo che ha portato allo sviluppo di tale funzionalità è legato al fatto che le soluzioni valutate nell'ambito di una singola esecuzione vengono generate casualmente. Ciò significa che, verosimilmente, in un'esecuzione successiva a quella portata a termine può aver individuato soluzioni migliori o peggiori, a parità di cardinalità. L'idea che nasce da tale considerazione, dunque, consiste nell'iterare un numero arbitrario di esecuzioni del software per poi fondere i risultati di ciascuna di essa, andando così a comporre un insieme finale di soluzioni con il meglio che ciascuna di esse può fornire. Sfruttando tale procedimento è possibile quantomeno avvicinarsi all'ottenimento degli ottimi globali, qualora non risultino già individuati, ed in generale si è rivelato utile per ottenere risultati migliori su dataset con un numero di topic elevato, dato che difficilmente una singola esecuzione ha modo di esplorare un gran numero di soluzioni di pari cardinalità, prima di terminare per via del limite imposto sul numero di iterazioni di *NSGA-II*.

[7] I valori delle opzioni `--et` (`--es`) e `--mx`.

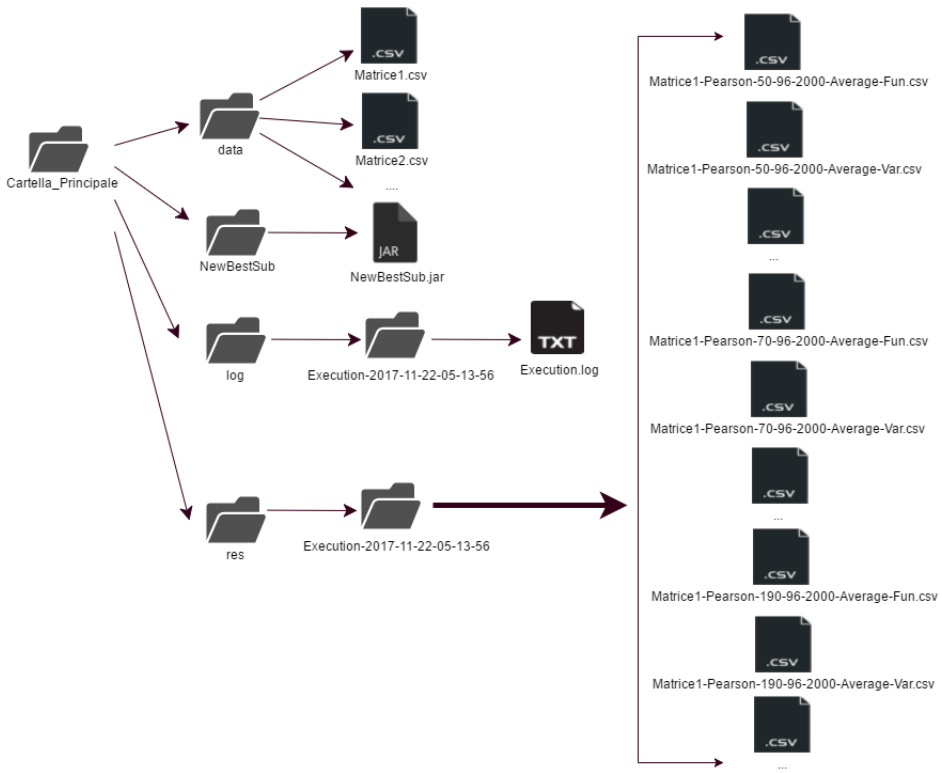


Figura 7.5: Organizzazione dei file sul file system al termine dell'uso della funzionalità di espansione dei topic di *NewBestSub*.

Per migliorare i risultati finali l'utente ha la possibilità di indicare come opzione da riga di comando il numero di esecuzioni da svolgere e delle quali fondere i risultati stessi[8]. Il relativo sottoprocesso consiste, come indicato nel paragrafo precedente, nell'iterare l'esecuzione del sistema più volte, con una singola differenza rispetto al comportamento standard. Tale differenza consiste nell'espandere la convenzione utilizzata per denominare i file prodotti da ciascuna delle esecuzioni aggiungendo un identificatore numerico per distinguere ciascuna delle esecuzioni iterate ai parametri già utilizzati. Una volte terminate tali esecuzioni e prodotti i relativi risultati, viene svolta la fase di analisi vera e propria. Tale fase consiste nella scansione riga per riga, contemporaneamente, dei file contenenti l'aggregazione dei risultati per ciascuna delle esecuzioni e nella creazione di un nuovo insieme di file etichettati aggiungendo ai nomi utilizzati dalla convezione precedentemente descritta un suffisso `Merged`, il quale indica che essi contengono l'esito della fase di analisi, che ha come output i

[8]Il valore dell'opzione `--mr`.

c	Be.	Wo.	Av.
1	.78	.77	.89
2	.80	.79	.88
3	.79	.92	.98

c	Be.	Wo.	Av.
1	.80	.84	.97
2	.55	.66	.43
3	.82	.89	.88

c	Be.	Wo.	Av.
1	.80	.77	.97
2	.80	.66	.88
3	.82	.89	.98

Tabella 7.2: Esempio di confronto effettuato durante l'operazione di miglioramento dei risultati.

risultati migliorati e definitivi. Durante la scansione, per ciascuna cardinalità vengono confrontati i valori finali di correlazione di ogni esperimento ed una volta dedotto quale esecuzione ha il valore migliore, tutte le informazioni disponibili relative alla soluzione contenuta nell'esecuzione così individuata, come i valori delle funzioni obiettivo, la composizione dei topic che la caratterizzano e quant'altro, vengono copiate nei nuovi file denominati con la nuova convenzione espansa con l'aggiunta del suffisso `Merged`. Come ultimo passo, i risultati delle singole esecuzioni iterate vengono eliminati.

Per comprendere meglio le operazioni di confronto dei valori di correlazione descritte nel paragrafo precedente, si osservino le tabelle dell'esempio 7.2 e si supponga che le prime due rappresentino i risultati ottenuti in due esecuzioni del sistema, per tutti e tre gli esperimenti. Nella terza tabella è possibile osservare l'effetto del processo di analisi dei risultati, una volta portato a termine. All'interno di tale tabella, dunque, è possibile individuare i migliori sottoinsiemi di topic che le due esecuzioni sono state in grado di fornire.

A questo punto, si supponga di eseguire il programma con la configurazione di opzioni da riga di comando dell'esempio 7.2 dove, in aggiunta, viene richiesto a *NewBestSub* di migliorare i risultati finali iterando tre esecuzioni, ottenendo la configurazione descritta nel listato 7.4. Una volta terminata l'operazione di miglioramento dei risultati, la situazione sul file system rappresentata nella figura 7.2 diventa quella rappresentata nella figura 7.6.

```
java -jar NewBestSub-1.0-jar-with-dependencies.jar -fi "matrice_1" -c "
    Pearson" -po 2000 -r 2000 -i 100000 -t "All" -pe 1,100 -l Limited -mr 3
```

Listato 7.4: Comando da terminale per iterare più esecuzioni di *NewBestSub* in modo da ottenere risultati finali migliori.

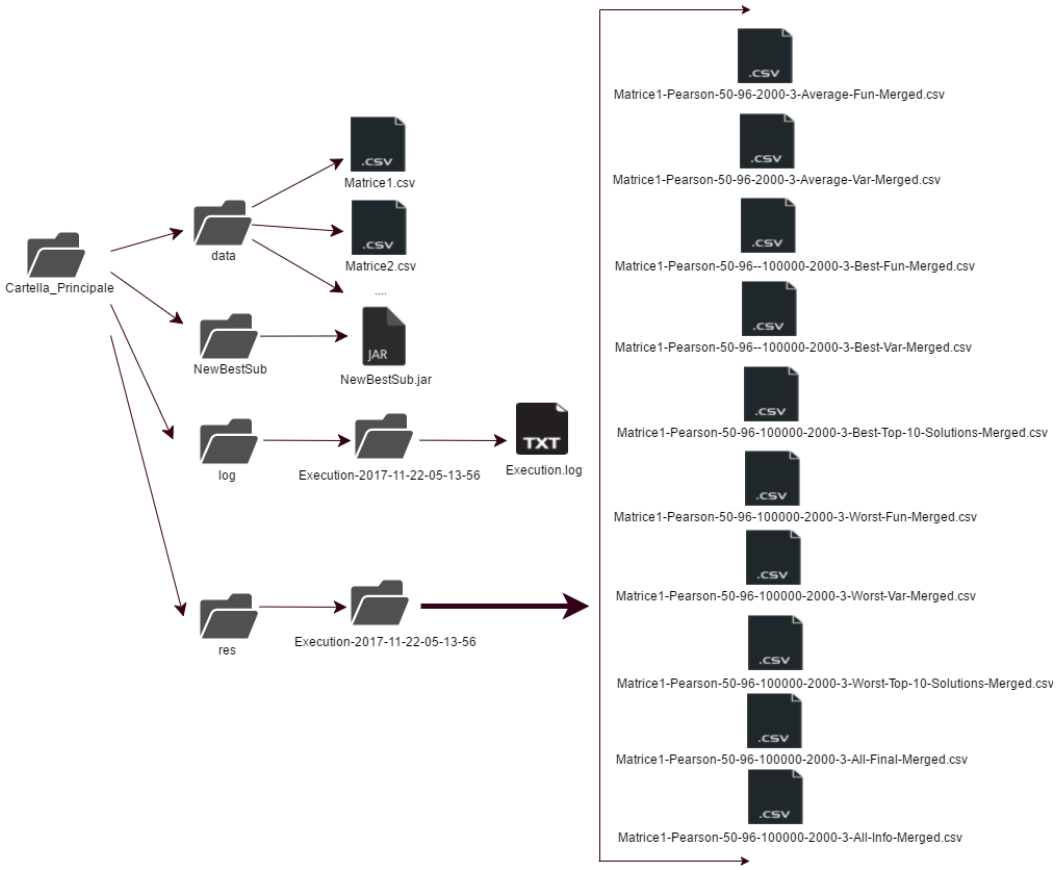


Figura 7.6: Organizzazione dei file sul file system al termine dell'uso della funzionalità di miglioramento dei risultati di *NewBestSub*.

7.2 Discussione

L'implementazione di *NewBestSub*, confrontata con quella di *BestSub*, è in grado di ottenere valori di correlazione migliori, ossia più *alti*, per l'esperimento *Best*, e più *bassi*, per l'esperimento *Worst*, come si vede nella sezione 8.2, per i dataset in input. Per ottenere tali risultati, vengono avviate diverse esecuzioni dello stesso *NewBestSub* (e, di conseguenza, di *NSGA-II)* dove ciascuna di esse analizza una diversa popolazione iniziale. Successivamente, grazie alla funzionalità di miglioramento dei risultati precedentemente descritta, i risultati finali di tali esecuzioni vengono fusi, ottenendo così i migliori sottoinsiemi individuati in assoluto. Tale pratica viene utilizzata comunemente nell'ambito degli *Algoritmi Evolutivi* e permettere di evitare una forma di *Overfitting* dei risultati stessi, oltre a permettere il miglioramento della loro qualità.

Più specificatamente, vengono lanciate dieci esecuzioni da fondere sui vari dataset in input e vengono comparati i valori di correlazione ottenuti da *NewBestSub* con quelli ottenuti da una singola esecuzione di *BestSub*. Quello si può notare svolgendo tale analisi è che si hanno piccoli miglioramenti, benché non decisivi, per dataset da 50 topic, mentre per quelli da più di 50 topic tali miglioramenti sono ben più consistenti e determinanti.

7.2.1 Limitazioni

NewBestSub, allo stato attuale, è caratterizzato da alcune limitazioni. Ad esempio, le componenti della versione 5.2 di *jMetal*, ossia quella effettivamente utilizzata nell'implementazione dello stesso *NewBestSub*, non sono in grado di garantire che per ogni cardinalità vengano individuate esattamente altre nove soluzioni. Ciò significa che gli insiemi delle migliori dieci soluzioni per ogni cardinalità possono avere dimensioni diverse. Ad ogni modo, sperimentalmente si vede come per dataset da cinquanta topic tale limitazione viene comunque evitata da *NewBestSub*, poiché riesce ad ottenere tali dieci soluzioni per ogni cardinalità, mentre per dataset più grandi ciò avviene comunque per la maggior parte delle cardinalità stesse. L'analisi del legame esistente fra il numero di soluzioni per ciascuna cardinalità che è possibile ottenere ed i parametri utilizzati nell'uso di *NSGA-II* viene posta come sviluppo futuro.

Un'ulteriore limitazione è dovuta alla versione 5.2 di *jMetal* consiste nel fatto che essa non permette in alcun modo di riprendere un'esecuzione interrotta per qualsiasi motivo, dall'errore umano all'esaurimento dello spazio su disco. Ciò può essere piuttosto problematico poiché per dataset con un numero elevato di topic l'esecuzione può durare diversi giorni. Per tale motivo, un'eventuale nuova versione di *NewBestSub* deve necessariamente considerare l'implementazione di una strategia di ripresa dell'esecuzione stessa.

Nell'implementazione vengono utilizzati i parametri allo stato dell'arte per gli operatori sfruttati da *NSGA-II* per tre motivi principali, ossia:

- evitare il rischio di *Overfitting* dovuto a tali parametri;
- fare in modo che l'algoritmo termini velocemente, senza necessariamente trovare i migliori sottoinsiemi *Best* e *Worst* in assoluto;
- mantenere l'implementazione semplice.

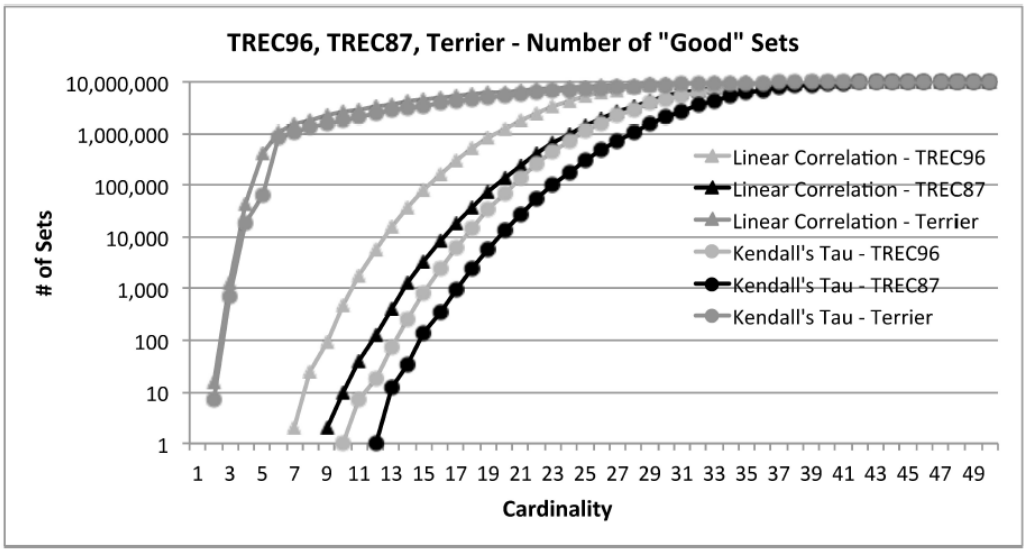


Figura 7.7: Numero di sottoinsiemi di topic "buoni" per ogni cardinalità di AH99-Top96 ed altre due collezioni, per tre popolazioni di sistemi di IR, tratta da Berto et al. [4, Figura 3].

A supporto della seconda osservazione, in particolare, vi sono in risultati di Berto et al. [4], i quali mostrano come tenda sempre ad esistere un grande numero di sottoinsiemi "buoni" di topic. In particolare, nell'articolo si mostra come per sottoinsiemi di cardinalità pari alla metà di quella originale generati analizzando il dataset AH99-Top96 descritto nella sezione 8.1, più del 50% di essi abbiano un valore di correlazione molto vicino ad uno, mentre con una cardinalità di almeno trentacinque topic le percentuali di tali sottoinsiemi *buoni* diventa il 99%, come si può vedere nella figura 7.7. Un'eventuale nuova versione di *NewBestSub*, dunque, dovrebbe prendere in considerazione la possibilità di poter regolare manualmente i parametri degli operatori sfruttati da *NSGA-II* al fine di poterne studiare gli effetti.

7.2.2 Tecnologie

Nella fase di implementazione di *NewBestSub* sono state utilizzate diverse tecnologie, non solo per quanto riguarda la programmazione, ma anche per l'attività di progettazione. Per quanto riguarda la seconda attività, nel corso di questa tesi sono stati presentati diversi schemi che, nella maggior parte dei casi, consistono in diagrammi delle classi o dei casi d'uso. Tali diagrammi sono stati costruiti basandosi

sullo standard *UML* e durante la composizione degli stessi ci si è affidati alle linee guida descritte in Fowler [18].

Per quando riguarda la fase di programmazione vera è propria, la prima scelta svolta è stata quella relativa al linguaggio da utilizzare. Poiché *jMetal* è stato sviluppato in *Java*, la scelta è parsa obbligata fin dall'inizio e, pertanto, durante le prime giornate di lavoro l'implementazione è andata avanti con tale linguaggio. Dopo tale fase iniziale sono emersi alcuni elementi che hanno portato ad una revisione di tale scelta. Negli ultimi anni, infatti, sono stati creati diversi linguaggi di programmazione orientata agli oggetti con elementi funzionali ed assunzioni interne che vincolano il programmatore in determinati modi costringendolo, ad esempio, a gestire esplicitamente i casi in cui una variabile può assumere valore nullo, al fine di garantire la *Null Safety* ed evitare l'incorrere in una `NullPointerException` o simili. Inoltre, più banalmente, molti dei linguaggi più recenti consentono al programmatore di scrivere codice meno verboso, ad esempio eliminando la necessità di inserire il punto e virgola per indicare la fine di uno *statement*. In sostanza, vi erano due fattori contrastanti che consistevano nella necessità di avere una forma di compatibilità con *jMetal* ed il desiderio di sfruttare le possibilità offerte da tali linguaggi. Tali considerazioni hanno portato alla scelta di *Kotlin*[9] come linguaggio di programmazione da utilizzare per sviluppare *NewBestSub*.

Kotlin è un linguaggio di programmazione orientato agli oggetti sviluppato dalla *JetBrains* che ha, come caratteristica principale, quello essere interoperabile al 100% con la *JVM*. Tale caratteristica è di grande importanza perché consente di utilizzare le classi contenute in qualsiasi programma diffuso mediante formato `.jar` come *jMetal* ed, in generale, di importare qualsiasi classe *Java*, interagendo con esse mediante la sintassi di *Kotlin* stesso. Nella prima metà del 2017 *JetBrains* ha stretto un accordo con *Google* e *Kotlin* è diventata *first-class language* per lo sviluppo su piattaforma *Android*[10]. Nei primi giorni di novembre dello stesso anno, inoltre, *Jetbrains* ha annunciato la possibilità di compilare programmi scritti in *Kotlin* direttamente in linguaggio macchina, evitando così l'uso della *JVM*. Tale linguaggio, dunque, è supportato piuttosto intensamente e tali aperture verso altre piattaforme ne hanno ampliato notevolmente le possibilità d'uso. Una delle possibilità più interessanti di *Kotlin* è quella di implementare in un modo piuttosto semplice una forma di programmazione parallela ed asincrona attraverso il meccanismo delle *Corouti-*

[9] `https://kotlinlang.org/`

[10] `https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/`

nes[11], le quali risparmiano al programmatore molte complicazioni che utilizzando *Java* è necessario gestire esplicitamente. Pur essendo allo stato attuale delle cose una funzionalità ancora sperimentale, tali *Coroutines* si sono rivelate sufficientemente stabili per implementare lo svolgimento in parallelo degli esperimenti *Best*, *Worst* ed *Average* all'interno del sistema. Elizarov [14], in particolare, fornisce una spiegazione piuttosto ampia di come esse possono essere utilizzate e di cos'è possibile ottenere. Inoltre, in rete è possibile trovare diverse pagine di confronto fra *Kotlin* ed altri linguaggi, incluso quello ufficiale svolto dalla stessa *JetBrains* con *Java*[12], e diversi articoli di sviluppatori entusiasti di tale linguaggio[13]. L'uso di *Kotlin* come linguaggio di programmazione per l'implementazione di *NewBestSub*, dunque, ha permesso di soddisfare entrambe le esigenze menzionate nel paragrafo precedente.

Un'ulteriore necessità che è risultato necessario soddisfare durante la fase d'implementazione è legata alla gestione delle dipendenze di *NewBestSub*, tra le quali vi sono le librerie di *Kotlin*, la quale è consistita nella ricerca di un componente che gestisse automaticamente il recupero delle versioni aggiornate di tali dipendenze, evitando di svolgere il tutto manualmente, con il rischio di inconsistenze tra versioni. La scelta finale è ricaduta su *Maven*, sistema di build sviluppato dalla *Apache Software Foundation*, il quale consente di specificare tutte le dipendenze richieste dal progetto ed i relativi legami esistenti fra di esse in un unico file con estensione `.pom`, chiamato *Project Object Model*. Tali dipendenze sono disponibili in determinati repository ed il sistema effettua automaticamente il download delle stesse da essi, scaricandole in locale o in un ulteriore repository centralizzato. Questo permette di recuperare in modo uniforme i vari file e di poter spostare il progetto da un ambiente all'altro in maniera indipendente, avendo la sicurezza di utilizzare sempre le stesse versioni delle librerie specificate nel *POM*.

Le dipendenze gestite tramite *Maven* e sfruttate nel corso dell'implementazione di *NewBestSub* sono di vario genere. Ad esempio, per la produzione e la lettura di tutti i file `.csv` è stata utilizzata *OpenCSV*[14], mentre per le funzionalità di logging e la gestione delle opzioni da riga di comando sono state utilizzate, rispettivamente, *log4j2*[15] e *Commons CLI*[16]. Anche le due librerie precedentemente indicate sono

[11] `https://kotlinlang.org/docs/reference/coroutines.html`

[12] `https://kotlinlang.org/docs/reference/comparison-to-java.html`

[13] `https://medium.com/@octskyward/why-kotlin-is-my-next-programming-language-c25c001e26e3`

[14] `http://opencsv.sourceforge.net/`

[15] `https://logging.apache.org/log4j/2.x/`

[16] `https://commons.apache.org/proper/commons-cli/`

progetti della *Apache Software Foundation*.

Nel capitolo 8 vengono descritti gli esperimenti svolti ed analizzati i risultati ottenuti eseguendo *NewBestSub* su diversi dataset in input. Infine, vengono brevemente discusse le tecnologie utilizzate nell'attività di analisi dei risultati stessi.

Capitolo 8

Esperimenti

Lo scopo di questo capitolo è presentare l'esito degli esperimenti svolti sui risultati ottenuti dalle esecuzioni di *NewBestSub*. In particolare, nella sezione 8.1 vengono elencati e descritti tutti i dataset successivamente analizzati. Successivamente, nella sezione 8.2 viene riprodotto il lavoro di Guiver et al. [20] per verificare l'efficacia dello stesso *NewBestSub* ed i miglioramenti ottenuti dal punto di vista dell'efficienza della computazione. Nella sezione 8.3, inoltre, viene descritta la fase di generalizzazione di tali risultati, secondo diversi aspetti. Infine, nella sezione 8.4 vengono svolti esperimenti per ottenere nuovi risultati i quali estendano gli studi di Guiver et al. [20].

8.1 Dataset

Negli esperimenti descritti in questo capitolo vengono utilizzati diversi dataset di *TREC*, i quali vengono riassunti nella tabella 8.1:

1. `AH99`: ottenuto utilizzando l'intera collezione del task *Ad Hoc* di *TREC* con AP come metrica, prendendo in considerazione tutti i sistemi.

2. `AH99-Top96`: ottenuto selezionando da `AH99` solamente i 96 sistemi più efficaci, ossia circa il 75% di essi (tale dataset è quello utilizzato da Guiver et al. [20]).

3. `AH99-LogAP`: ottenuto utilizzando $LogAP$ come metrica (tale metrica consiste nel calcolare il logaritmo di ciascuno dei valori di AP per un dataset nella forma visibile nella figura 2.1).

4. `AH99-LogAP-Top96`: ottenuto selezionando da `AH99-LogAP` solamente i 96 sistemi più efficaci, ossia circa il 75% di essi.

5. `AH99-AP@20`: ottenuto utilizzando uno *shallow pool* (ossia, i valori di AP vengono calcolati prendendo in considerazione solamente i primi 20 documenti reperiti, i quali vengono successivamente valutati).

6. `AH99-AP@20-Top96`: ottenuto selezionando da `AH99-AP@20` solamente i 96 sistemi più efficaci, ossia circa il 75% di essi.

7. `WEB14`: ottenuto utilizzando l'intera collezione del track *Web* del task *Ad Hoc* di *TREC* del 2014 con *NDCG* come metrica e prendendo in considerazione tutti i sistemi.

8. `WEB14-Top25`: ottenuto selezionando da `WEB14` solamente i 25 sistemi più efficaci, ossia circa il 75% di essi.

9. `WEB14B-0-0`: ottenuto svolgendo una prima forma di "binarizzazione" (descritta nel seguito) dei valori categoriali che caratterizzano i giudizi di relevance al fine di poter calcolare valori di AP come metrica.

10. `WEB14B-0-0-Top25`: ottenuto selezionando da `WEB14B-0-0` solamente i 25 sistemi più efficaci, ossia circa il 75% di essi.

11. `WEB14B-01-0`: ottenuto svolgendo una seconda forma di "binarizzazione" (anch'essa descritta nel seguito) dei valori categoriali che caratterizzano i giudizi di relevance al fine di poter calcolare valori di AP come metrica.

12. `WEB14B-01-0-Top25`: ottenuto selezionando da `WEB14B-01-0` solamente i 25 sistemi più efficaci, ossia circa il 75% di essi.

13. `TB06`: ottenuto utilizzando l'intera collezione del track *Terabyte* di *TREC* del 2006 con AP come metrica e prendendo in considerazione tutti i sistemi.

14. `TB06-Top49`: ottenuto selezionando da `TB06` solamente i 49 sistemi più efficaci, ossia circa il 75% di essi.

15. `R04`: ottenuto utilizzando l'intera collezione del track *Robust* di *TREC* del 2004, con AP come metrica e prendendo in considerazione tutti i sistemi.

16. `R04-Top82`: ottenuto selezionando da `R04` solamente gli 82 sistemi più efficaci, ossia circa il 75% di essi.

17. `MQ07-Top26`: ottenuto utilizzando l'intera collezione del track *Million Query* di *TREC* del 2007, AP come metrica e selezionando solamente i 26 sistemi più efficaci, ossia circa il 75% di essi.

#	Acronimo	Nome Ufficiale	Anno	Num. Topic	Num. Sistemi
1	AH99	Ad Hoc Task	1999	50	129
2	AH99-Top96	Ad Hoc Task	1999	50	96
3	AH99-LogAP	Ad Hoc Task	1999	50	129
4	AH99-LogAP-Top96	Ad Hoc Task	1999	50	96
5	AH99-AP@20	Ad Hoc Task	1999	50	129
6	AH99-AP@20-Top96	Ad Hoc Task	1999	50	96
7	WEB14	Web Track	2014	50	30
8	WEB14-Top25	Web Track	2014	50	25
9	WEB14B-0-0	Web Track	2014	50	30
10	WEB14B-0-0-Top25	Web Track	2014	50	25
11	WEB14B-01-0	Web Track	2014	50	30
12	WEB14B-01-0-Top25	Web Track	2014	50	25
13	TB06	Terabyte Track	2006	149	61
14	TB06-Top49	Terabyte Track	2006	149	49
15	R04	Robust Track	2004	249	110
16	R04-Top82	Robust Track	2004	249	82
17	MQ07-Top26	Million Query Track	2007	1153	26

Tabella 8.1: Collezioni utilizzate negli esperimenti.

L'utilizzo di WEB14 ha permesso di analizzare una collezione di test più recente e di effettuarne la comparazione con i risultati ottenuti durante i workshop di *TREC-8*. Tale aspetto è importante poiché i risultati originali di Guiver et al. [20] sono stati ottenuti analizzando una collezione piuttosto datata. Inoltre, ciò ha permesso di eseguire *NewBestSub* su una collezione con una metrica per l'efficacia diversa da *AP*, poiché WEB14 utilizza *NDCG* per tale scopo. Come si vede nella sezione 2.1, tale metrica richiede di scegliere uno tra diversi gradi di relevance per ciascun documento. Ciascun giudizio di relevance (*qrels*), dunque, è categoriale ed, in questo caso, assume un valore appartenente all'insieme $\{-2, 0, 1, 2, 3\}$. Il significato che assume ciascuna delle cinque categorie a cui fanno riferimento tali gradi di relevance viene descritto nella tabella 8.2.

Lo scopo del dataset WEB14B è quello di "binarizzare" i qrel di WEB14 in modo da poter calcolare dei valori di AP e ottenere un dataset nella forma riportata

Etichetta	Grado	Descrizione
Key	3	La pagina è dedicata al topic; autorevole e comprensibile; è degna di essere tra i risultati migliori in un motore di ricerca.
HRel	2	Il contenuto della pagina fornisce informazioni sostanziali relative al topic.
Rel	1	Il contenuto della pagina fornisce alcune informazioni relative al topic, le quali possono essere minimali; le informazioni pertinenti devono essere presenti in tale pagina.
Non	0	Il contenuto della pagina non fornisce informazioni utili relative al topic, ma può fornire informazioni utili relative ad altri topic, incluse altre interpretazioni dell'interrogazione.
Junk	-2	La pagina non sembra utile ad alcuno scopo; potrebbe essere spazzatura o spam.

Tabella 8.2: Significato dei cinque gradi di relevance utilizzati in WEB14, tratto da Collins-Thompson et al. [8, pp. 6–7].

nella figura 2.1. In particolare, vengono svolte due forme distinte di binarizzazione. Nel dataset WEB14B-0-0 i qrel $\in \{-2, 0\}$ vengono considerati come non pertinenti, mentre quelli $\in \{1, 2, 3\}$ vengono considerati pertinenti. Ciò significa che per il calcolo dei valori di AP i primi assumono un valore pari a 0, mentre i secondi un valore pari a 1. Per quanto riguarda il dataset WEB14B-01-0, i qrel $\in \{-2, 0, 1\}$ vengono considerati come non pertinenti, mentre quelli $\in \{2, 3\}$ vengono considerati pertinenti.

L'analisi svolta nel seguito si focalizza sugli effetti dati dall'uso di metriche diverse per la valutazione, dall'uso della tecnica del pooling, dalla profondità dei pool ottenuti e dall'analisi di collezioni di test diverse. Per quanto riguarda il numero dei topic, durante l'analisi di ciascuna collezione tale parametro non subisce alcuna modifica, poiché lo studio degli effetti legati ad eventuali sue variazioni viene posto come sviluppo futuro.

8.2 Riproduzione dei Risultati Precedenti

Lo scopo dei primi esperimenti svolti consiste nella riproduzione dei risultati originali ottenuti da Guiver et al. [20] al fine di poter verificare l'efficacia dell'implementazione di *NewBestSub* ed i miglioramenti ottenuti per quanto riguarda l'efficienza della computazione.

8.2.1 Efficacia

Per quanto riguarda la verifica dell'efficacia di *NewBestSub*, l'esperimento consiste nel confrontare i valori di correlazione che caratterizzano i sottoinsiemi di topic *Best*, *Worst* ed *Average* individuati per ogni cardinalità con quelli ottenuti da *BestSub*. Tale esperimento, dunque, viene svolto sui dataset per i quali esistono dei risultati individuati da Guiver et al. [20], per entrambi i metodi utilizzati per calcolare i valori di correlazione. Essi, in particolare, sono AH99-Top96 (Figura 8.1), TB06-Top49 (Figura 8.2), R04-Top82 (Figura 8.3) e MQ07-Top26 (Figura 8.4).

In generale, si può notare come le serie di valori di correlazione individuate da *NewBestSub* e da *BestSub* risultino praticamente indistinguibili per la maggior parte dei dataset. Ad esempio, prendendo in considerazione i valori di correlazione calcolati mediante τ di Kendall per AH99-Top96 (Figura 8.1b), *NewBestSub* individua valori diversi da quelli di *BestSub* solamente per cardinalità attorno alla 12-esima ed alla 31-esima per la serie *Best*, ed attorno alla 40-esima per la serie *Worst*. La serie *Average* risulta stabile con 10000 ripetizioni e si sovrappone perfettamente con quella originale, ottenuta con 250 ripetizioni.

L'esperimento *Worst* risulta quello più difficile da riprodurre poiché soggetto a maggiori variazioni dal punto di vista dei valori di correlazione calcolati, specialmente quando ciò viene svolto mediante τ di Kendall. Non sempre, però, tali variazioni risultano indesiderate. Se il sottoinsieme di topic individuato per una cardinalità c ha un valore di correlazione maggiore (per gli esperimenti *Best* ed *Average*) o minore (per l'esperimento *Worst*) rispetto a quello individuato da *BestSub*, allora tale variazione assume una connotazione positiva ed indica che *NewBestSub* è più efficace di *BestSub*. Particolarmente interessanti, da questo punto di vista, sono i risultati ottenuti per MQ07-Top26 (Figura 8.4). Infatti, si può notare come per i valori calcolati mediante ρ di Pearson nell'ambito dell'esperimento *Worst* (Figura 8.4a), vi sia un deciso miglioramento per circa 160 cardinalità. In altre parole, i valori calcolati da *NewBestSub* sono molto più bassi di quelli calcolati da *BestSub*. Per tale dataset, inoltre, *BestSub* riesce ad individuare sottoinsiemi di topic sola-

mente per le prime 250 cardinalità su 1153 (in due mesi), mentre *NewBestSub* riesce ad analizzarle tutte in poche ore. *NewBestSub*, dunque, non solo riesce a riprodurre correttamente i risultati individuati, ma anche ad ottenerne di nuovi da dataset molto più grandi di quelli utilizzati da Guiver et al. [20].

8.2.2 Efficienza

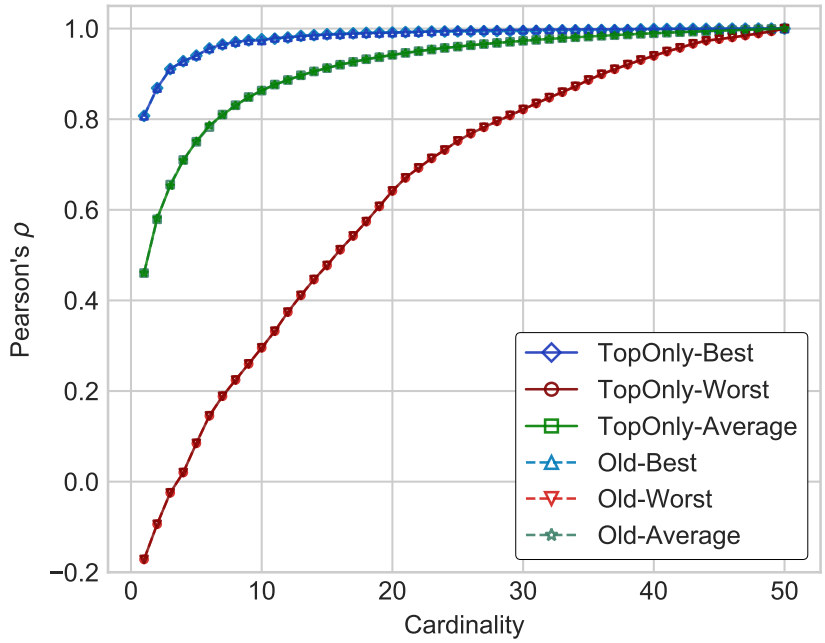
Avendo dimostrato che *NewBestSub* è in grado di riprodurre e migliorare i risultati originali ottenuti da Guiver et al. [20] mediante *BestSub*, è possibile analizzarne l'efficienza dal punto di vista delle due dimensioni del problema, ossia il numero di topic ed il numero di sistemi del dataset in input. A tale scopo, vengono svolti due esperimenti distinti, i quali consistono in:

1. svolgere più esecuzioni del software fissando il numero dei topic ed aumentando di volta in volta quello dei sistemi;

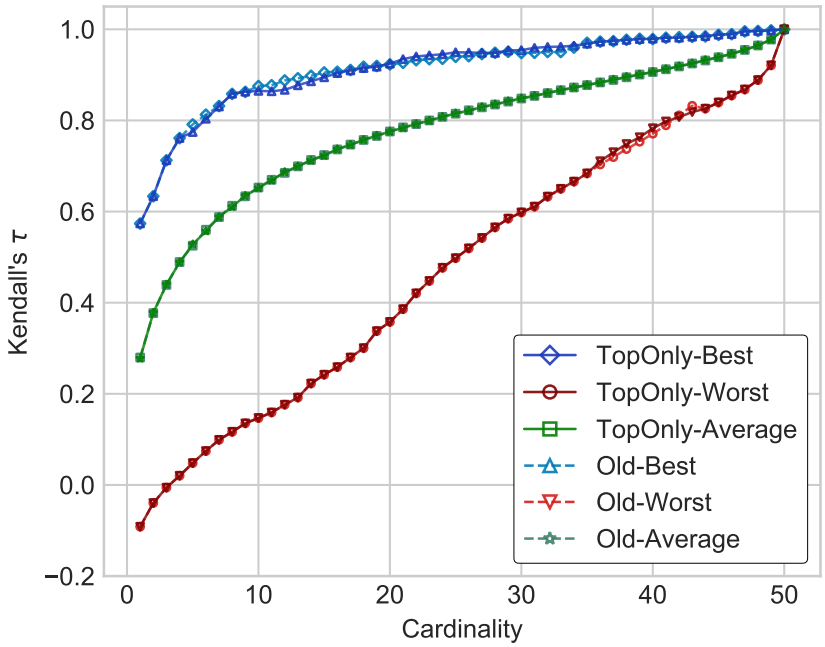
2. svolgere più esecuzioni del software fissando il numero dei sistemi ed aumentando di volta in volta quello dei topic.

Gli esperimenti descritti vengono svolti sfruttando le funzionalità di espansione dei sistemi o dei topic descritte nella sottosezione 7.1.5. Per quanto riguarda il primo di tali esperimenti, il numero dei topic viene fissato a 50 mentre quello dei sistemi, per la prima esecuzione, a 5. Successivamente, ne vengono aggiunti altri 5 di esecuzione in esecuzione, fino ad arrivare ad un totale pari a 96 (tali parametri, in particolare, sono quelli che caratterizzano il dataset AH99-Top96 il quale, in questo caso, viene utilizzato come *benchmark*). Per il secondo esperimento, il numero dei sistemi viene fissato a 96 mentre quello dei topic, per la prima esecuzione, a 50. Similmente al caso precedente, ne vengono aggiunti altri 5 di esecuzione in esecuzione, fino ad arrivare ad un totale pari a 1150 (tale parametro, in particolare, consiste in un'approssimazione del numero di topic che caratterizza il più grande dataset disponibile, ossia MQ07-Top26).

Gli effetti dell'aumento del numero di sistemi m sulla computazione di *NewBestSub* possono essere compresi analizzando i risultati per il primo esperimento, riportati nella tabella 8.3. In particolare, più tale numero m aumenta, più i vettori utilizzati per il calcolo dei valori di correlazione sono lunghi. La complessità finale, dunque, è $O(m)$ quando tali valori vengono calcolati mediante ρ di Pearson, mentre è $O(m \log m)$ se vengono calcolati mediante τ di Kendall. Ad ogni modo, l'impatto sulla computazione è piuttosto ridotto e l'andamento della complessità temporale risulta poco più che lineare, con una crescita contenuta.

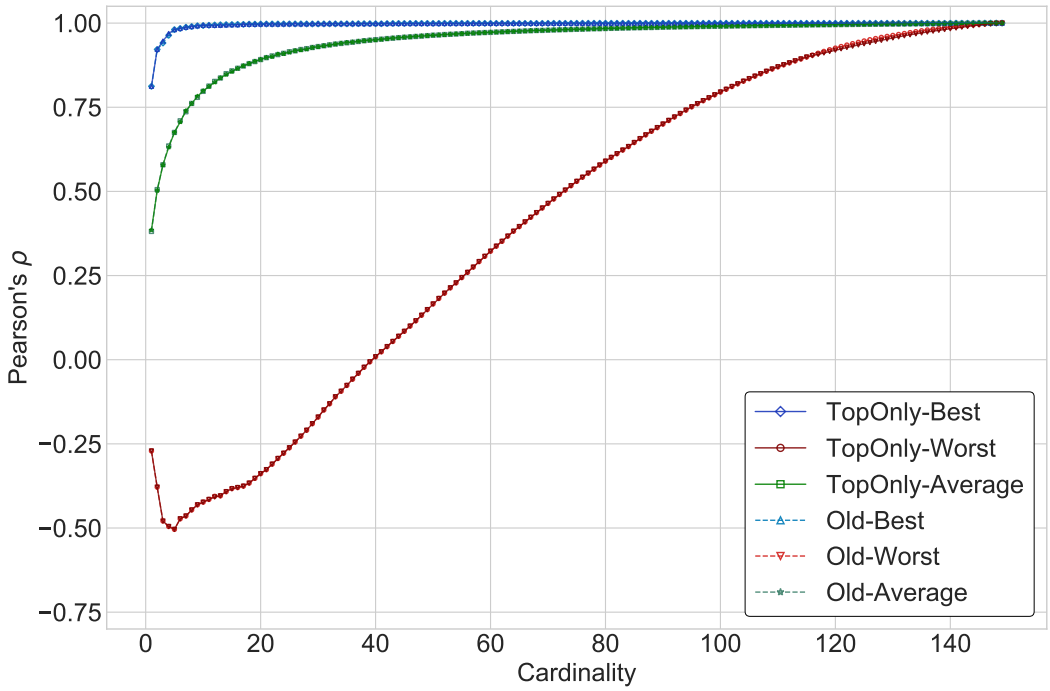


(a) Valori di correlazione calcolati mediante ρ.

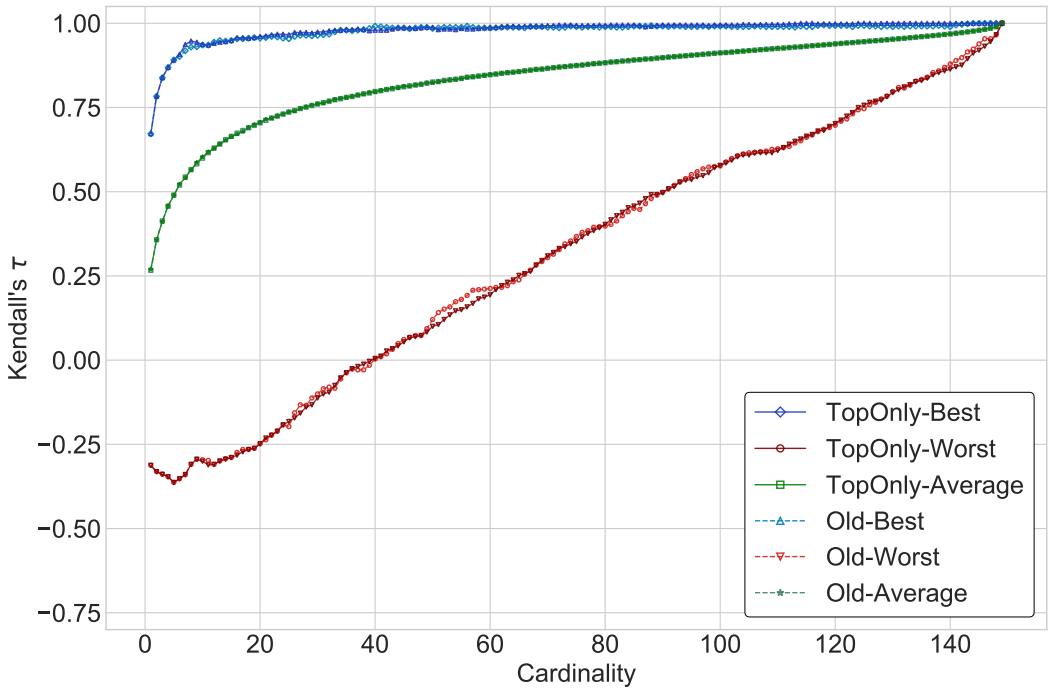


(b) Valori di correlazione calcolati mediante τ.

Figura 8.1: Confronto tra i valori di correlazione ottenuti da *BestSub* e quelli ottenuti da *NewBestSub* per il dataset AH99-Top96.

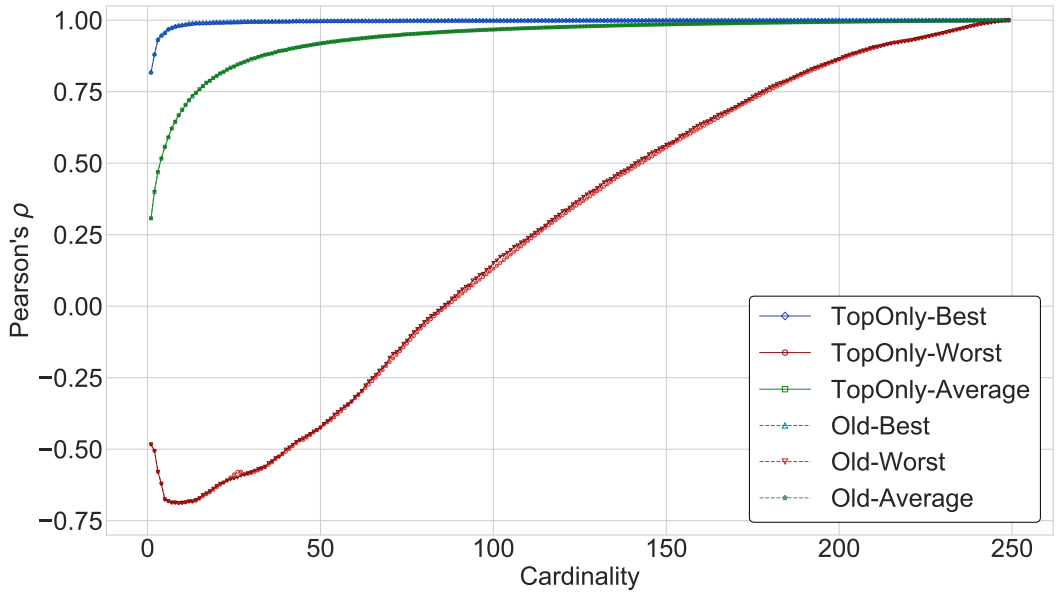


(a) Valori di correlazione calcolati mediante ρ.

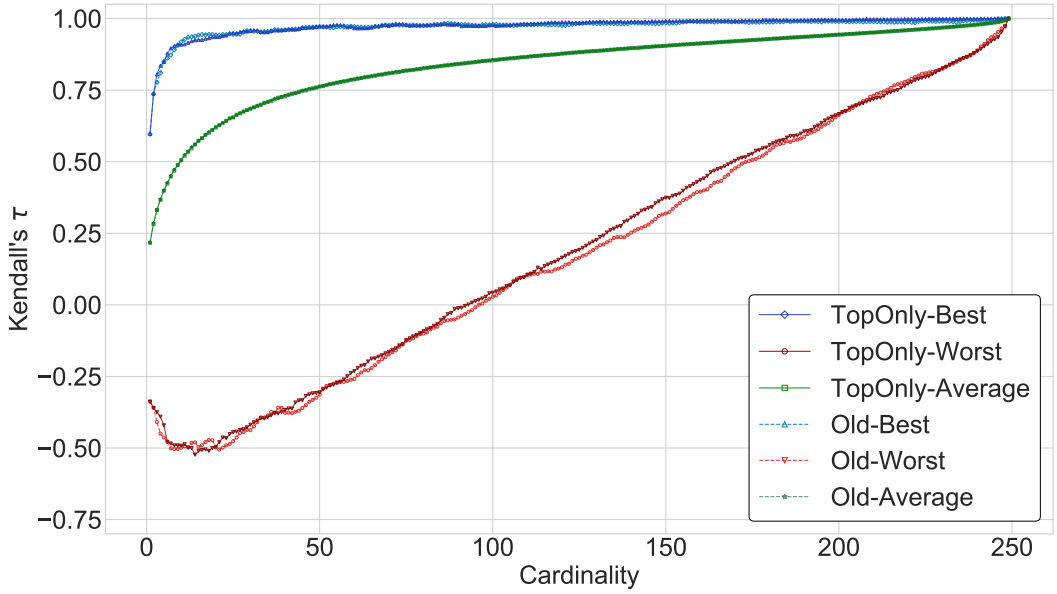


(b) Valori di correlazione calcolati mediante τ.

Figura 8.2: Confronto tra i valori di correlazione ottenuti da *BestSub* e quelli ottenuti da *NewBestSub* per il dataset TB06-Top49.

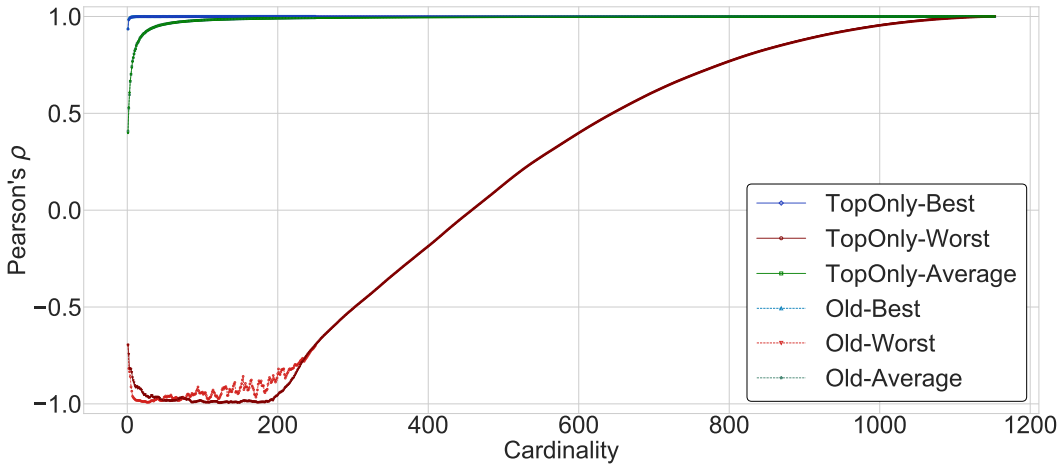


(a) Valori di correlazione calcolati mediante ρ.

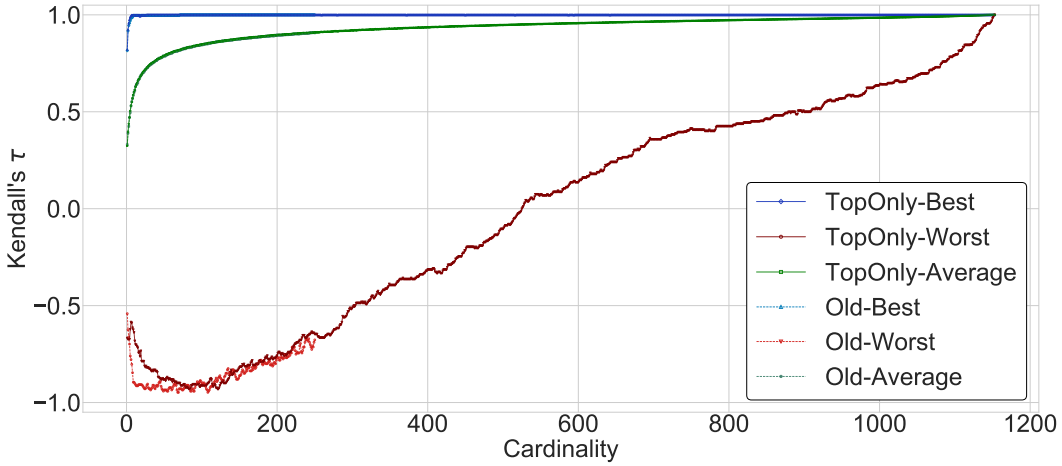


(b) Valori di correlazione calcolati mediante τ.

Figura 8.3: Confronto tra i valori di correlazione ottenuti da *BestSub* e quelli ottenuti da *NewBestSub* per il dataset R04-Top82.



(a) Valori di correlazione calcolati mediante ρ.



(b) Valori di correlazione calcolati mediante τ.

Figura 8.4: Confronto tra i valori di correlazione ottenuti da *BestSub* e quelli ottenuti da *NewBestSub* per il dataset MQ07-Top26.

Num. Topic	Num. Sistemi	Tempo *NewBestSub*
50	5	2.00 Min
50	10	2.06 Min
50	25	2.14 Min
50	40	2.25 Min
50	50	2.30 Min
50	75	2.50 Min
50	90	2.63 Min
50	96	2.69 Min

Tabella 8.3: Misurazione della durata di alcune esecuzioni di *NewBestSub* svolte variando il numero dei sistemi.

Num. Topic	Num. Sistemi	Tempo *NewBestSub*	k	Tempo *BestSub*	Speedup
50	96	3 Min	2	30 Min	10x
50	96	3 Min	3	12 Ore	240x
250	96	10 Min	2	1 Mese	4380x
250	96	10 Min	3	$>$ 4 Mesi*	$>$ 17,520x
500	96	20 Min	2	$>$ 1 Anno*	$>$ 26,280x
750	96	35 Min	2	$>$ 1 Anno*	$>$ 15,017x
1000	96	60 Min	2	$\gg$ 1 Anno*	$\gg$ 8760x
1100	96	80 Min	2	$\gg$ 1 Anno*	$\gg$ 6570x

(*) L'esecuzione è stata fermata prima del termine.

Tabella 8.4: Comparazione temporale di alcune esecuzioni di *BestSub* e *NewBestSub* effettuata al variare del numero dei topic.

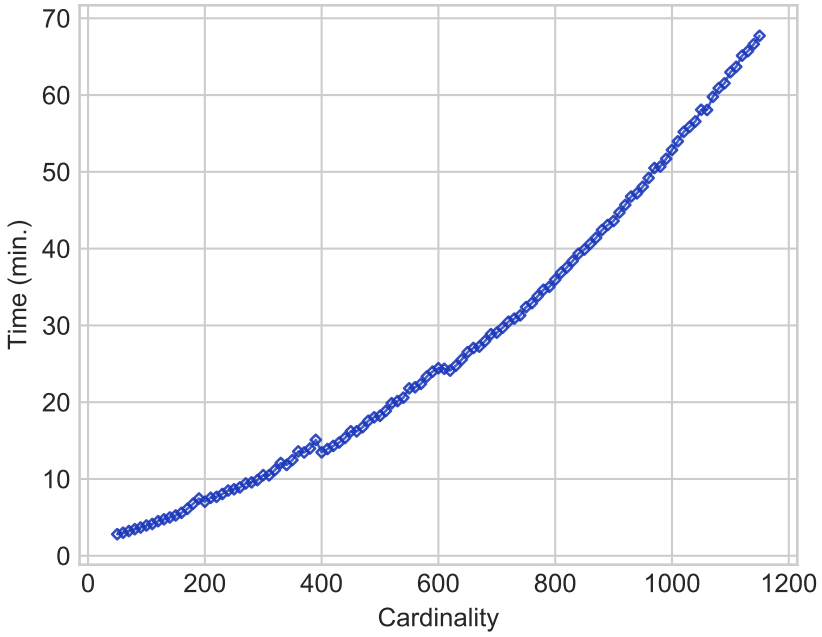
Gli effetti dell'aumento del numero dei topic sulla computazione di *NewBestSub* possono essere compresi analizzando i risultati per il secondo esperimento, riportati nella tabella 8.4. Come si può notare, tale dimensione del problema influenza la durata della computazione ben più drasticamente del numero dei sistemi e, perciò, risulta essere il fattore chiave per l'efficienza del software. L'andamento della complessità temporale risulta, anche in questo caso, più che lineare, con una crescita decisamente più alta rispetto a quella data dall'aumento del numero dei sistemi la quale, tuttavia, non risulta esponenziale come quella di *BestSub*. Tale aspetto si può notare osservando le figure 8.5, le quali permettono di visualizzare graficamente l'andamento del tempo di computazione all'aumentare del numero di topic (Figura 8.5a) ed un confronto di tale andamento con il dato disponibile relativo all'esecuzione di *BestSub* (Figura 8.5b). Da tale confronto emerge come un esecuzione di *BestSub* con un numero di topic pari a 50 necessiti, per terminare, di circa dieci volte il tempo impiegato da un'esecuzione di *NewBestSub* con un numero di topic pari a 1150. In altre parole, un'esecuzione di *BestSub* sul più piccolo dataset disponibile è meno efficiente di un'esecuzione di *NewBestSub* su quello più grande disponibile.

Le tre colonne di destra della tabella 8.4 analizzano più dettagliatamente il fenomeno descritto nella figura 8.5b. In particolare, per comparare l'efficienza di *BestSub* con quella di *NewBestSub* viene calcolato lo *Speedup* ottenuto dal secondo, secondo la seguente formula:

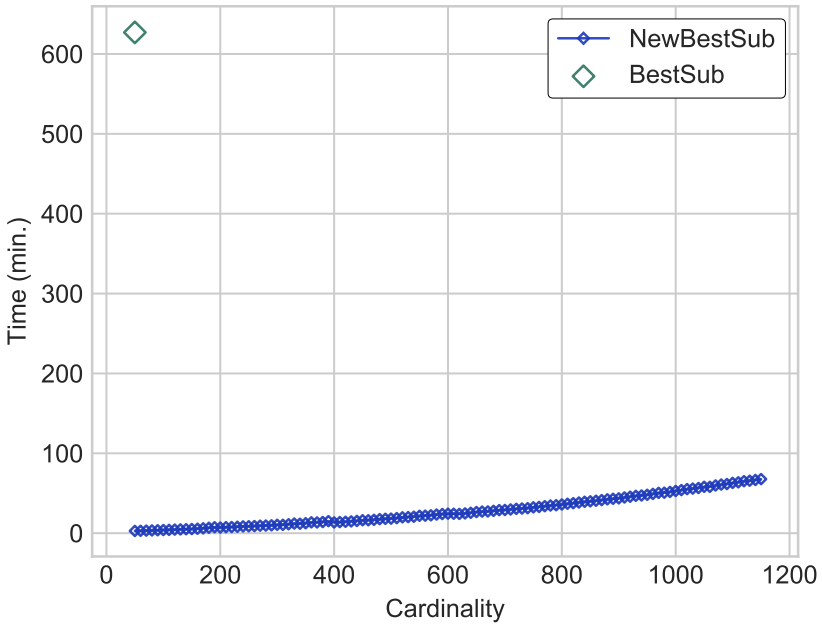
$$\text{Speedup} = \frac{\text{Tempo}(BestSub)}{\text{Tempo}(NewBestSub)} \tag{8.1}$$

In generale, si nota come lo speedup ottenuto da *NewBestSub*, anche quando *BestSub* viene utilizzato con un valore basso per il parametro k al fine di velocizzare l'euristica, sia decisamente consistente. Nonostante il tempo di computazione venga drasticamente ridotto, la qualità dei risultati non viene influenzata, poiché i valori di correlazione ottenuti dai due software sono simili, almeno fino a quando tale confronto è possibile, ossia fino a 250 topic. *BestSub*, infatti, non è risultato in grado di andare oltre a tale limite per via dell'inefficienza della sua computazione.

NewBestSub, inoltre, è molto più efficace ed efficiente anche quando analizza dataset con un numero elevato di topic. Ad esempio, si osservino la terza e la quarta riga della tabella 8.4. In tali casi vengono lanciate 10 esecuzioni del software per massimizzare/minimizzare il più possibile i valori di correlazione dei sottoinsiemi *Best*/*Worst*, per un dataset da 96 sistemi e 250 topic. Il tempo totale impiegato da *NewBestSub* è pari a 10 esecuzioni per 10 minuti ad esecuzione, per un totale di 100 minuti. I valori di correlazione ottenuti da esso risultano molto più bassi per i sottoinsiemi *Worst* se confrontati con quelli individuati da *BestSub*, dove esso viene



(a) Andamento del tempo di computazione.



(b) Confronto con *BestSub*.

Figura 8.5: Efficienza di *NewBestSub* all'aumentare del numero dei topic.

eseguito con un valore per il parametro k della sua euristica pari a 2 (per quelli calcolati mediante τ di Kendall, ad esempio, vi è una differenza massima pari a 0.2). *BestSub*, inoltre, viene eseguito anche con un valore per k pari a 3, per calcolare valori di correlazione solamente fino a cardinalità 50, avendo comunque a disposizione dati per 250 cardinalità (ottenuti con circa un mese e mezzo di computazione). Anche in questo caso, i risultati ottenuti da *NewBestSub* con 10 esecuzioni sono significativamente migliori, specialmente per i sottoinsiemi *Worst*.

L'efficienza molto più elevata di *NewBestSub* non è solamente teorica/temporale, poiché ha reso possibile effettuare proficuamente nuovi esperimenti con diversi fini, i quali vengono discussi nelle sezioni 8.3 e 8.4.

8.3 Generalizzazione dei Risultati

Lo scopo degli esperimenti descritti nel seguito è quello di generalizzare i nuovi risultati ottenuti utilizzando *NewBestSub* secondo diversi aspetti quali:

- un confronto tra lo svolgimento dell'analisi di un dataset includendo tutti i sistemi o solo una percentuale di quelli più efficaci;
- un'analisi delle stabilità dei sottoinsiemi *Best*/*Worst* individuati per ciascuna cardinalità;
- un'analisi delle stabilità generalizzate ai migliori dieci sottoinsiemi *Best*/*Worst* individuati per ciascuna cardinalità;
- un'analisi degli effetti dati dall'uso di una collezione di test più recente (la quale, inoltre, sfrutta una metrica per l'efficacia diversa, ossia *NDCG*) e dall'uso di uno *shallow pool*

8.3.1 Dataset Non-Top vs. Dataset Top

Uno dei problemi che ostacola la riproduzione di risultati originali consiste nel non utilizzare un dataset nella sua interezza (Ferro et al. [16] e Ferro [15]). L'esperimento svolto per analizzare tale aspetto consiste nel confrontare l'esito delle esecuzioni effettuate includendo tutti i sistemi di un dato dataset con quello delle esecuzioni effettuate includendo solamente una percentuale dei sistemi più efficaci, la quale viene solitamente posta al 75%. Tale esperimento viene effettuato tra i dataset originali ed una loro versione che includa solamente tale percentuale di

sistemi più efficaci, denominata seguendo la convenzione "NomeDatasetOriginale-TopNumeroSistemi", quando presente (ad esempio, per AH99 vi è AH99-Top96). Nel seguito viene presentato un confronto per ciascuna tipologia di dataset (ossia, uno per ciascun task/track di *TREC*). In particolare, vi sono sono AH99 vs. AH99-Top96 (Figura 8.6), TB06 vs. TB06-Top49 (Figura 8.7), R04 vs. R04-Top82 (Figura 8.8) e WEB14 vs. WEB14-Top25 (Figura 8.10).

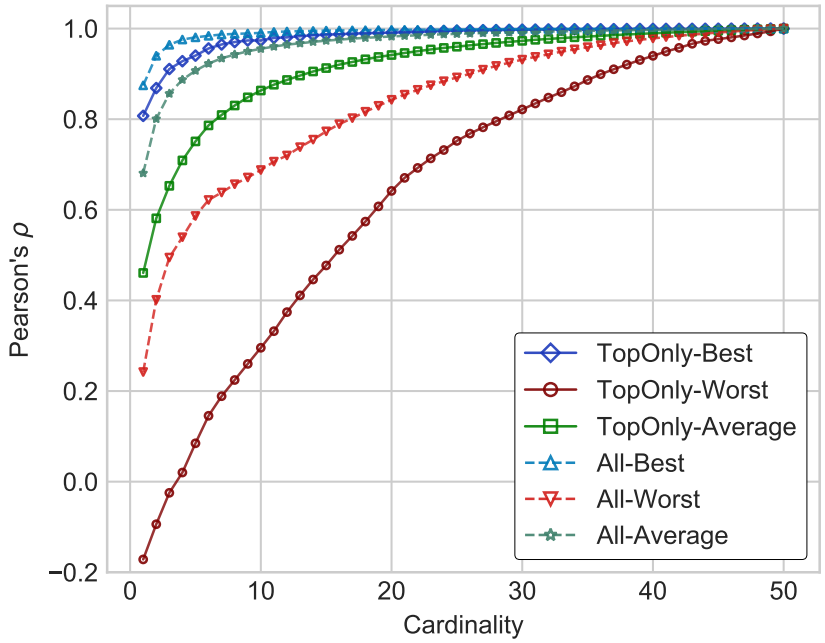
Il fatto di utilizzare solamente il 75% dei sistemi più efficaci porta ad ottenere risultati decisamente diversi rispetto a quando vengono utilizzati tutti. Per quanto riguarda la serie *Worst*, i valori di correlazione risultano minori di quelli ottenuti nel secondo caso, per la maggior parte dei dataset. Ad esempio, considerando i valori di correlazione calcolati mediante τ di Kendall, si ottiene una correlazione pari a 0.4 con 8 topic per il dataset AH99-Top96, mentre considerando l'intero dataset AH99 ne servono 21. Per il dataset TB06-Top49, si ottiene una correlazione pari a 0.4 con 44 topic, mentre considerando l'intero dataset TB06 ne servono 80.

Per quanto riguarda le serie *Best* ed *Average*, al contrario, i valori di correlazione risultano maggiori considerando tutti i sistemi anziché il 75% di quelli più efficaci, benché la forbice sia molto meno ampia rispetto a quella del caso precedente. Prendendo in considerazione i dataset descritti negli esempi precedenti ed i valori di correlazione calcolati mediante τ di Kendall per la serie *Best*, si ottiene una correlazione pari a 0.9 con 9 topic per il dataset AH99, mentre per AH99-Top96 ne servono 16. Per quanto riguarda TB06, si ottiene una correlazione pari a 0.9 con 4 topic, mentre per TB06-Top49 ne servono 6. Per quanto riguarda la serie *Average*, le considerazioni sono analoghe. Inoltre, quanto descritto nei paragrafi precedenti risulta valido per tutte e tre le serie anche quando i valori di correlazione vengono calcolati mediante ρ di Pearson.

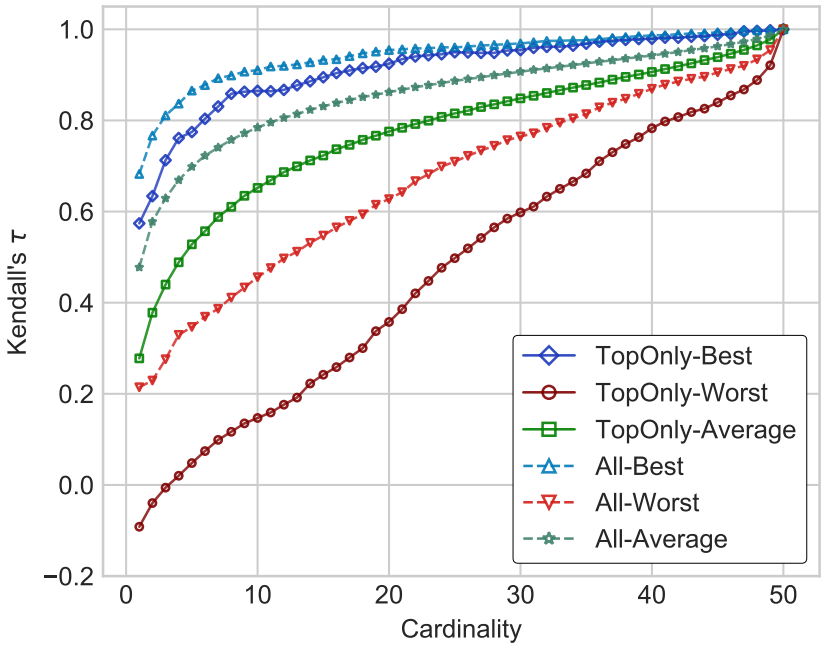
8.3.2 Stabilità dei Sottoinsiemi *Best/Worst*

Nella sezione 2.3 viene descritto il modo in cui l'euristica utilizzata da *BestSub* può alterare i risultati di stabilità dei sottoinsiemi *Best/Worst* che esso individua. *NewBestSub* si basa un'euristica diversa (e più sofisticata) e, per tale motivo, è risultato necessario svolgere un nuovo studio relativo alla stabilità dei sottoinsiemi individuati dallo stesso *NewBestSub* per verificare eventuali effetti causati da tale euristica e se l'uso di quella precedente, utilizzata da *BestSub*, può apportare comunque dei benefici.

Le figure 8.9, in particolare, mostrano due pixel-map costruite per analizzare la stabilità dei sottoinsiemi *Best/Worst* individuati da *NewBestSub* per il dataset

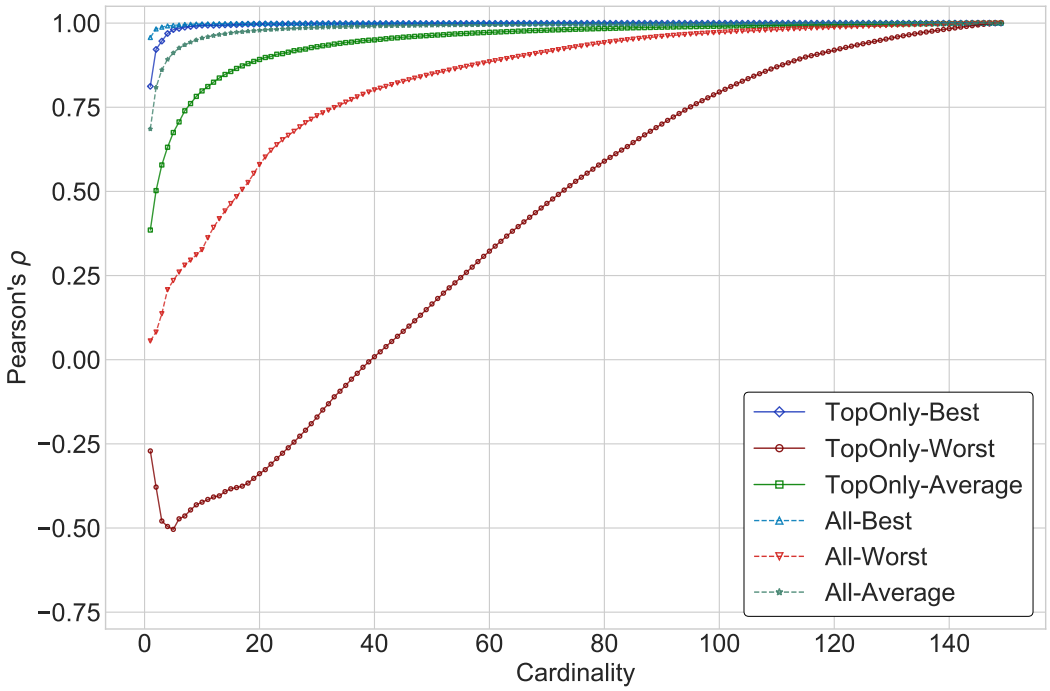


(a) Valori di correlazione calcolati mediante ρ.

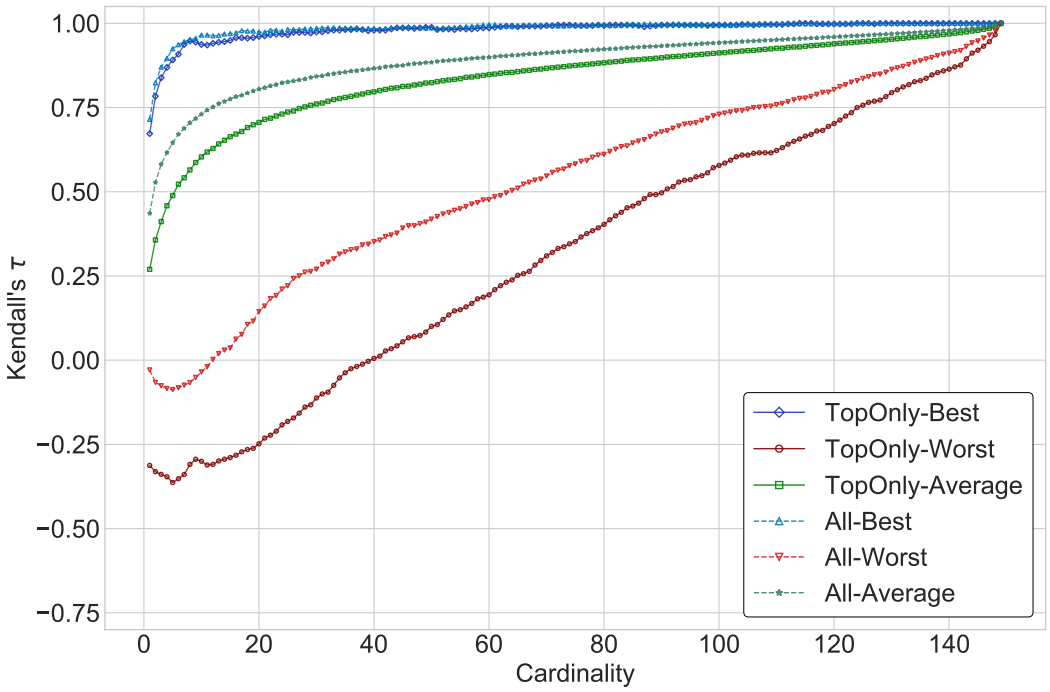


(b) Valori di correlazione calcolati mediante τ.

Figura 8.6: Confronto tra i valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti per i dataset AH99/AH99-Top96.

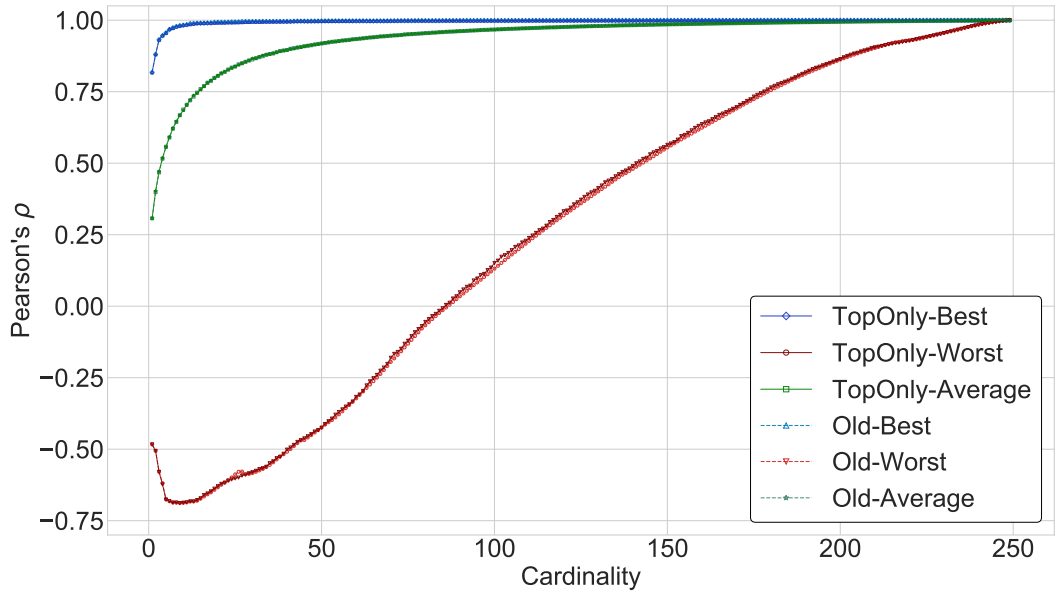


(a) Valori di correlazione calcolati mediante ρ.

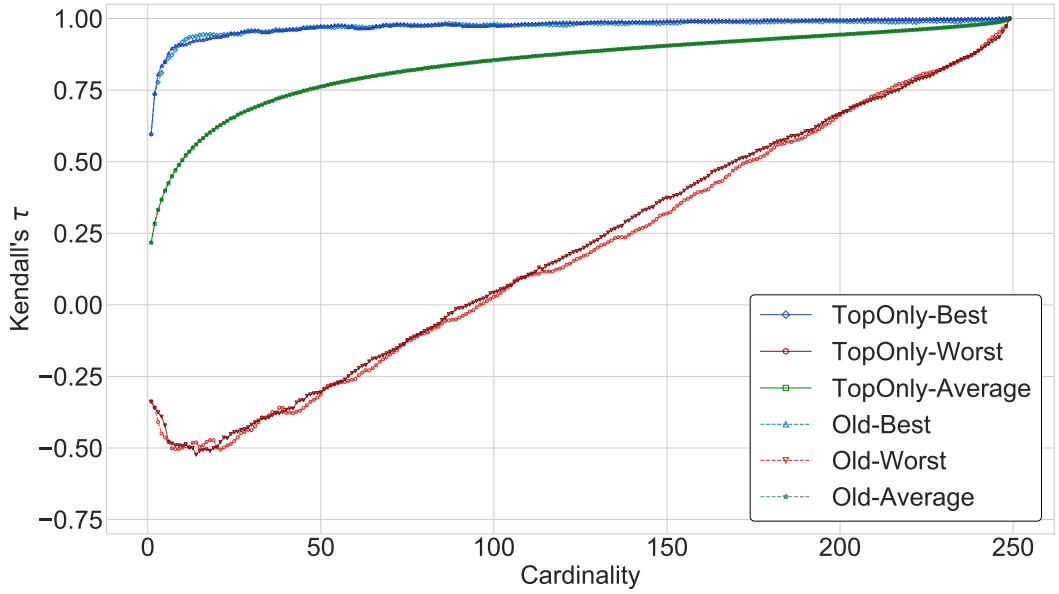


(b) Valori di correlazione calcolati mediante τ.

Figura 8.7: Confronto tra i valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti per i dataset TB06/TB06-Top49.

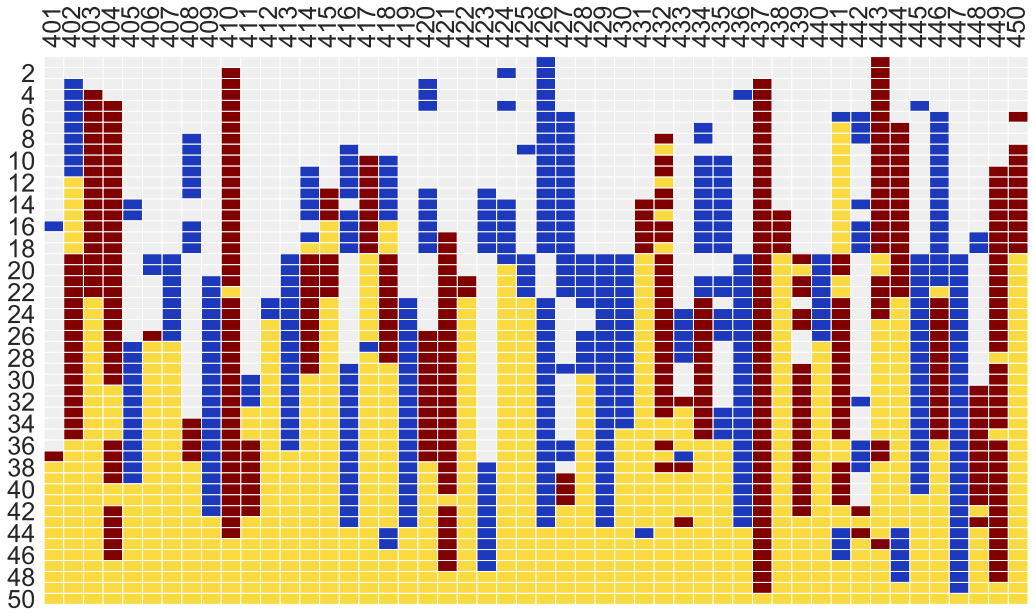


(a) Valori di correlazione calcolati mediante ρ.

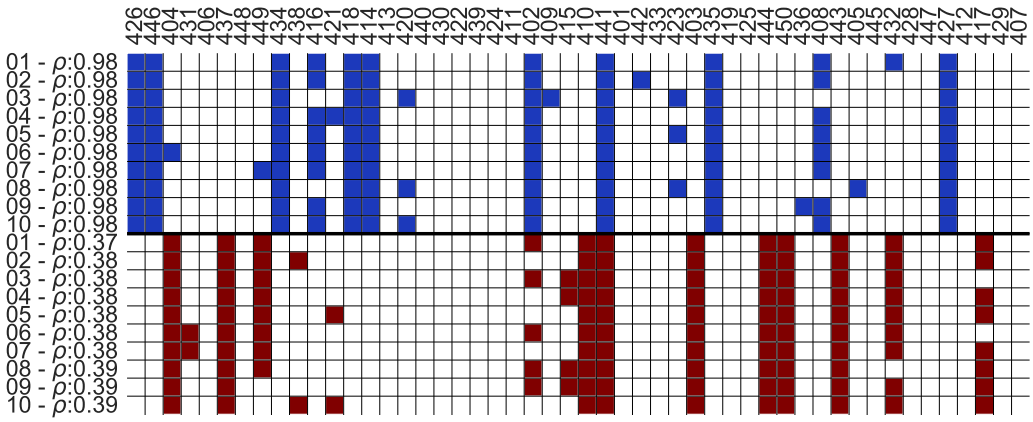


(b) Valori di correlazione calcolati mediante τ.

Figura 8.8: Confronto tra i valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti per i dataset R04/R04-Top82.



(a) Pixel-map per l'intero dataset.



(b) Pixel-map per i migliori 10 sottoinsiemi di cardinalità 12.

Figura 8.9: Pixel-maps costruite per analizzare la stabilità dei sottoinsiemi *Best/Worst* individuati da *NewBestSub*, con valori di correlazone calcolati mediante ρ per il dataset AH99-Top96.

AH99-Top96, dove i valori di correlazione vengono calcolati mediante ρ di Pearson. Il significato di tali figure è analogo quello delle figure 2.6, con la sola differenza data dall'uso del colore blu al posto del verde. Tale colore, in particolare, viene utilizzato per rappresentare i topic appartenenti ai sottoinsiemi *Best*, mentre il rosso quelli appartenenti ai sottoinsiemi *Worst*. Infine, il giallo indica topic appartenenti ad entrambe le tipologie di sottoinsiemi. Ogni colonna della pixel-map visibile nella figura 8.9a, dunque, descrive la stabilità di un dato topic nell'ambito dei sottoinsiemi *Best*/*Worst* individuati. Per quanto riguarda la pixel-map vibisile nella figura 8.9b, le dieci righe della sezione superiore della mostrano la composizione dei migliori dieci sottoinsiemi *Best* di cardinalità 12, mentre le dieci righe della sezione inferiore mostrano la composizione dei migliori dieci sottoinsiemi *Worst*.

Quando le pixel-map costruite analizzando l'output di *NewBestSub* vengono confrontate con quelle ottenute dall'uso di *BestSub* la stabilità dei sottoinsiemi *Best* e dei migliori 10 di essi risulta maggiore per *NewBestSub*, mentre quella dei sottoinsiemi *Worst* ed i migliori 10 di essi risulta simile per entrambi i software. Per verificare tale ipotesi, la stabilità dei sottoinsiemi *Best*/*Worst* individuata da *NewBestSub* viene misurata secondo il procedimento definito da Guiver et al. [20, p. 14], per il quale:

> Per ogni topic in un sottoinsieme di una data dimensione, un contatore viene incrementato se il topic corrente compare anche nel sottoinsieme di cardinalità $c + 1$. Per la ricerca esaustiva, tale contatore ha un valore minimo ed un valore massimo, dove quello minimo è maggiore di 0 poiché mano a mano che i sottoinsiemi di topic diventano più grandi inevitabilmente si verificano alcune sovrapposizioni tra di essi. Il valore di stabilità (espresso come percentuale) è dato dal rapporto calcolato mediante la seguente formula (8.3.2):
>
> $$\frac{\text{valore attuale del contatore} - \text{valore minimo del contatore}}{\text{valore massimo del contatore} - \text{valore minimo del contatore}}$$
>
> Per i sottoinsiemi *Worst* la stabilità media dei valori di *Average Precision* per tre misurazioni è pari al 93%. I sottoinsiemi *Best* sono, in qualche modo, meno consistenti, con una stabilità media pari all'86%.

Si noti che Guiver et al. [20] non specificano come calcolare il valore massimo e minimo di tale contatore. Le relative formule vengono derivate come segue, dove n

è il numero dei topic:

$$\text{valore mimino del contatore} = \begin{cases} 0 & \text{se } c \leq \lfloor \frac{n}{2} \rfloor - 1 \\ 2c + 1 - n, & \text{altrimenti} \end{cases}$$

$$\text{valore massimo del contatore} = c$$

La tabella 8.5 mostra i valori di stabilità calcolati per i sottoinsiemi *Best*/*Worst* ottenuti da *NewBestSub* per tutti i dataset analizzati (Tabella 8.1). Inoltre, vengono inclusi anche i valori calcolati da *BestSub* per AH99-Top96, come termine di paragone. Comparando le performance di *NewBestSub* e *BestSub* sullo stesso dataset (righe 0 e 2 della tabella 8.5), differentemente dall'ipotesi indotta dalle figure 8.9, i valori di stabilità risultano praticamente identici. Ciò risulta chiaro anche quando vengono comparate i valori di stabilità medi di tutti i dataset analizzati da *NewBestSub* (ultima riga della tabella) con quelli ottenuti da *BestSub* per AH99-Top96. Ciò che si nota è che i valori di correlazione risultano simili per tutti i dataset, con i valori di stabilità per i sottoinsiemi *Worst* maggiori di quelli per i sottoinsiemi *Best*. Ciò non è del tutto vero per le varianti di *WEB14B*, in quanto vi è un divario piuttosto ampio tra i valori stessi ma ciò può essere dovuto al processo di binarizzazione. Ad ogni modo, i risultati ottenuti confermano le considerazioni di Guiver et al. [20], ossia:

1. quando un insieme di topic entra in un sottoinsieme *Worst* ad una certa cardinalità tende a rimanervici anche per quelle seguenti;

2. la considerazione precedente è meno vera per insiemi di topic facenti parte di un sottoinsieme *Best*;

3. un sottoinsieme *Worst* viene costituito da topic *Worst* a livello individuale, mentre i sottoinsiemi *Best* non sono necessariamente formati da soli topic *Best* a livello individuale. In altre parole, un sottoinsieme formato da topic *Worst* a livello individuale è *Worst*, mentre un sottoinsieme formato da topic *Best* a livello individuale non è necessariamente *Best*.

Sulla base di tutto ciò è possibile concludere che i risultati di stabilità ottenuti da Guiver et al. [20] risultano validi anche con la nuova (e diversa) euristica utilizzata nell'implementazione di *NewBestSub* e, pertanto, è improbabile che siano strettamente dipendenti dall'uso di quella definita da Guiver et al. [20].

#	Dataset	ρ di Pearson		τ di Kendall	
		Best Set	*Worst* Set	*Best* Set	*Worst* Set
0	AH99-Top96 (*BestSub*)	.88	.98	.84	.95
1	AH99	.85	.97	.86	.97
2	AH99-Top96	.88	.98	.88	.95
3	AH99-LogAP	.83	.96	.85	.92
4	AH99-LogAP-Top96	.89	.95	.82	.93
5	AH99-AP@20	.83	.96	.85	.92
6	AH99-AP@20-Top96	.89	.94	.89	.90
7	WEB14	.88	.92	.84	.89
8	WEB14-Top25	.87	.95	.82	.88
9	WEB14B-0-0	.85	.97	.61	.91
10	WEB14B-0-0-Top25	.86	.99	.56	.88
11	WEB14B-01-0	.70	.98	.73	.86
12	WEB14B-01-0-Top25	.71	.97	.70	.92
13	TB06	.93	.97	.89	.95
14	TB06-Top49	.93	.98	.88	.94
15	R04	.95	.98	.91	.97
16	R04-Top82	.94	.98	.92	.95
17	MQ07-Top26	.92	.99	.48	.98
	Media	.87	.97	.79	.92

Tabella 8.5: Valori di stabilità per i sottoinsiemi *Best/Worst* calcolati utilizzando la formula di Guiver et al. [20], con valori di correlazione calcolati mediante ρ e τ.

8.3.3 Stabilità dei Migliori 10 Sottoinsiemi *Best/ Worst*

Alla luce dei risultati analizzando la stabilità dei sottoinsiemi *Best/ Worst* è possibile generalizzare tale analisi ai migliori 10 sottoinsiemi *Best/ Worst*, analogamente a quanto fatto da Guiver et al. [20] e da Berto et al. [4]. A tale scopo, la misura di stabilità descritta nella sottosezione 8.3.2 viene modificata in modo da poter prendere in considerazione i migliori p sottoinsiemi *Best/ Worst* per una data cardinalità c (in particolare, si ha $p = 10$). In altre parole, è necessario aggiornare le formule per il calcolo dei valori massimo e minimo del contatore riportate nella sottosezione 8.3.2, mentre il rapporto descritto dalla formula 8.3.2 ed utilizzato per il calcolo del valore finale di stabilità rimane immutato. Le formule aggiornate per il calcolo dei valori massimo e minimo del contatore, dunque, sono quelle definite da Guiver et al. [20], per le quali:

$$\text{valore minimo del contatore} = \begin{cases} (2c - n)(p - 1), & \text{se } c > \left\lfloor \frac{n}{2} \right\rfloor \\ 0 & \text{altrimenti} \end{cases}$$

$$\text{valore massimo del contatore} = c(p - 1)$$

La tabella 8.6 mostra i valori di stabilità per i migliori 10 sottoinsiemi *Best/ Worst* per alcune cardinalità, solamente per i valori di correlazione calcolati mediante τ di Kendall, poiché il risultato per la ρ di Pearson è simile. Come si può notare analizzando la tabella lungo ciascuna colonna (ossia, ciascuna cardinalità), i risultati sono simili per ciascun dataset, con la sola eccezione di WEB14 e WEB14B (e le sue varianti). Tale fenomeno, tuttavia, può essere dovuto al processo di binarizzazione che tali dataset subiscono, come già precisato nella sottosezione 8.3.2. Analizzando la tabella lungo ciascuna riga (ossia, ciascuna collezione di test), si osservano valori di stabilità piuttosto diversi per ciascuna cardinalità. Il valore massimo viene raggiunto generalmente attorno alle cardinalità 20 e 30, mentre il minimo viene raggiunto attorno alle cardinalità 5 e 45.

Ad ogni modo, la considerazione principale che si può trarre è che generalizzando i risultati descritti nella sottosezione 8.3.2 si nota come i migliori 10 sottoinsiemi *Worst* risultino decisamente più stabili dei migliori 10 sottoinsiemi *Best*. Inoltre, l'inclusione di tutti i sistemi di un dato dataset o del 75% di quelli più efficaci ha un effetto sui valori di stabilità. Osservando la tabella 8.6, infatti, si nota come l'utilizzo di tale percentuale dei sistemi più efficaci porta ad un peggioramento di tali valori di stabilità. Ad esempio, considerando cardinalità 5, la stabilità dei migliori dieci

#	Dataset	Migliori 10 *Best* Set						Migliori 10 *Worst* Set					
		Cardinalità						Cardinalità					
		5	10	20	30	40	45	5	10	20	30	40	45
1	AH99	.60	.74	.89	.92	.80	.65	.74	.84	.94	.93	.84	.71
2	AH99-Top96	.49	.82	.92	.90	.84	.64	.71	.84	.93	.91	.72	.67
3	AH99-LogAP	.58	.78	.88	.88	.85	.73	.73	.83	.91	.92	.88	.73
4	AH99-LogAP-Top96	.60	.81	.80	.86	.62	.62	.60	.86	.91	.88	.86	.71
5	AH99-AP@20	.54	.74	.90	.83	.62	.62	.67	.82	.90	.92	.78	.56
6	AH99-AP@20-Top96	.56	.77	.94	.89	.70	.56	.67	.82	.92	.89	.71	.51
7	WEB14	.67	.72	.91	.61	.53	.22	.73	.83	.92	.93	.86	.58
8	WEB14-Top25	.47	.64	.89	.46	.21	.20	.78	.82	.90	.89	.84	.62
9	WEB14B-0-0	.29	.20	.47	.43	.26	.17	.69	.85	.75	.89	.82	.60
10	WEB14B-0-0-Top25	.24	.57	.40	.39	.20	.05	.67	.79	.90	.84	.86	.31
11	WEB14B-01-0	.36	.59	.40	.56	.17	.99	.47	.80	.92	.87	.82	.60
12	WEB14B-01-0-Top25	.40	.50	.36	.52	.38	.35	.64	.82	.94	.86	.87	.71
13	TB06	.62	.86	.85	.96	.96	.93	.67	.83	.92	.95	.95	.96
14	TB06-Top49	.64	.79	.93	.91	.97	.90	.60	.83	.93	.94	.93	.93
15	R04	.71	.81	.94	.83	.95	.97	.69	.74	.92	.93	.96	.97
16	R04-Top82	.80	.84	.95	.96	.96	.95	.62	.90	.90	.90	.95	.96
	Media	.53	.70	.78	.74	.63	.60	.67	.83	.91	.90	.85	.70

Tabella 8.6: Valori di stabilità per i migliori 10 sottoinsiemi *Best*/*Worst*, con valori di correlazione calcolati mediante τ di Kendall.

sottoinsiemi *Best* per AH99 è pari a 0.60, mentre mentre per AH99-Top96 è pari a 0.49.

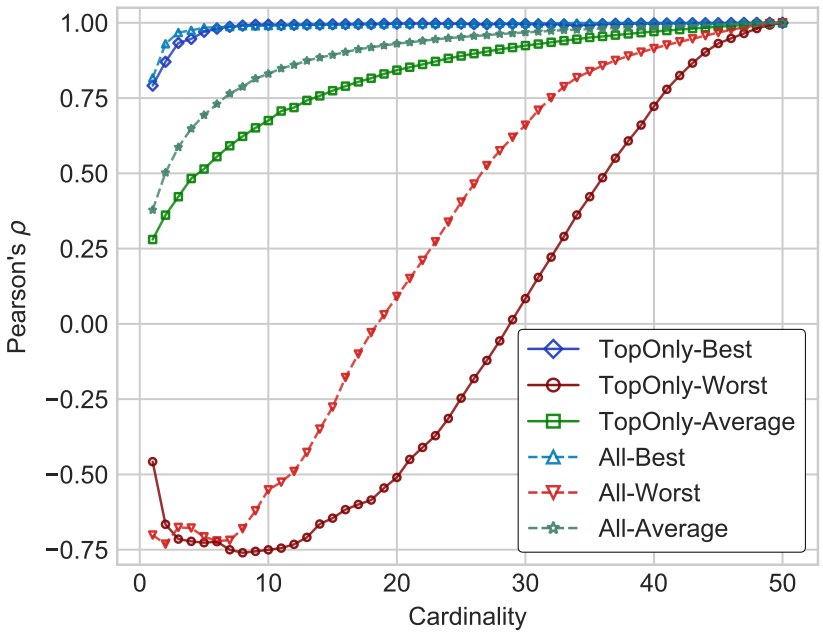
8.3.4 Una Collezione Più Recente con *NDCG* ed uno *Shallow Pool*

Tra le varie collezioni di test analizzate utilizzando *NewBestSub* ve ne sono due (con relative varianti) le quali vanno a definire dei casi particolari. La prima di esse è WEB14, la quale si distingue da quelle rimanenti poiché è molto più recente (il track *Web* risale al 2014, mentre il task "Ad Hoc" al 1999) ed usa una diversa metrica per la misurazione dell'efficacia dei sistemi di IR, ossia *NDCG*.

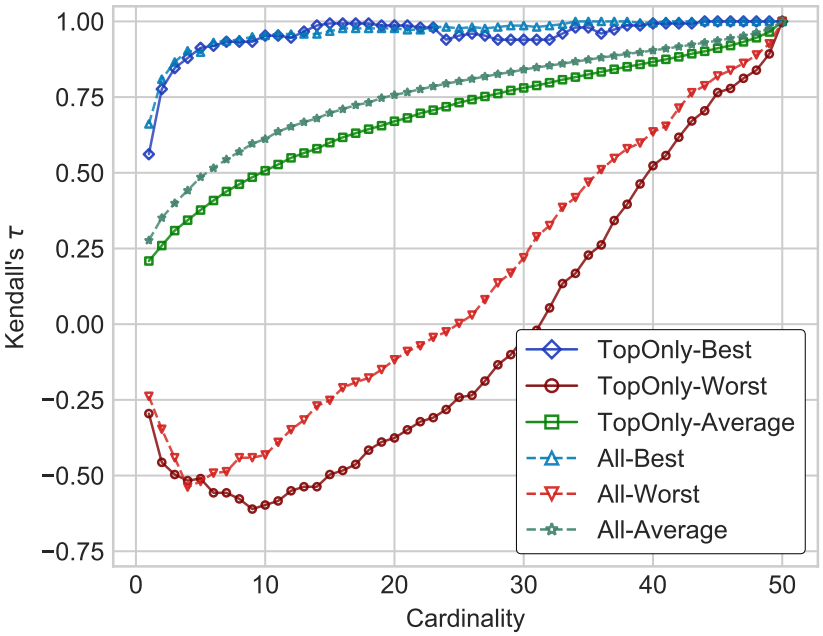
Per quanto riguarda WEB14, i risultati ottenuti sono simili a quelli già individuati in precedenza. In particolare, il fatto di includere tutti i sistemi o il 75% di quelli più efficaci nell'analisi porta ad ottenere valori di correlazione diversi ed il relativo confronto è visibile nei grafici contenuti nelle figure 8.10. Tale confronto ha un esito analogo a quello svolto per AH99 e visibile nei grafici contenuti nelle figure 8.6, in quanto le tre serie vengono caratterizzate dallo stesso andamento. Anche i risultati di stabilità (Tabella 8.5) sono allineati a quelli ottenuti per le altre collezioni. Per quanto riguarda quelli generalizzati ai migliori 10 sottoinsiemi *Best*/*Worst* essi, in alcuni casi, risultano peggiori delle altre collezioni e tali peggioramenti sono piuttosto consistenti per i sottoinsiemi *Best* di cardinalità 30, 40 e 45. Ad ogni modo, la considerazione che si può trarre dell'analisi dei dataset WEB14 e WEB14-Top25 è che i risultati ottenuti rimangono validi anche quando la collezione analizzata utilizza una metrica per l'efficacia dei sistemi di IR diversa.

Relativamente all'uso di uno *shallow pool*, confrontando i risultati ottenuti per AH99 (Figure 8.6) con quelli ottenuti per AH99-AP@20 (Figure 8.11), ossia il dataset derivante dall'applicazione di tale tecnica, si può notare come l'inclusione di tutti i sistemi o del 75% di quelli più efficaci nell'analisi abbia effetti sui sottoinsiemi *Worst*. In particolare, nella figura 8.11a si può osservare l'andamento dei valori di correlazione calcolati mediante ρ di Pearson all'aumentare della cardinalità. Per circa il 78% di esse risultano migliori i sottoinsiemi *Worst* individuati analizzando l'intero dataset, mentre utilizzando la τ di Kendall o prendendo in considerazione i sottoinsiemi *Best* i risultati sono analoghi a quelli ottenuti per AH99. Le considerazioni relative ai risultati di stabilità (Tabella 8.5) ed a quelli generalizzati ai migliori 10 sottoinsiemi *Best*/*Worst* (Tabella 8.6), infine, sono simili a quelle tratte per le altre collezioni.

Per concludere, è possibile affermare che i risultati riprodotti nella sezione 8.2 sono stabili anche con collezioni più recenti e facenti uso di *NDCG* come metrica



(a) Valori di correlazione calcolati mediante ρ.



(b) Valori di correlazione calcolati mediante τ.

Figura 8.10: Confronto tra i valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti per i dataset WEB14B/WEB14B-Top25.

per l'efficacia dei sistemi di IR e/o di uno shallow pool, benché utilizzando quest'ultima tecnica vi siano effetti sui valori di correlazione indipendenti dalla stabilità dei sottoinsiemi individuati, come quelli visibili analizzando i grafici in figura 8.11a.

8.4 Espansione dei Risultati

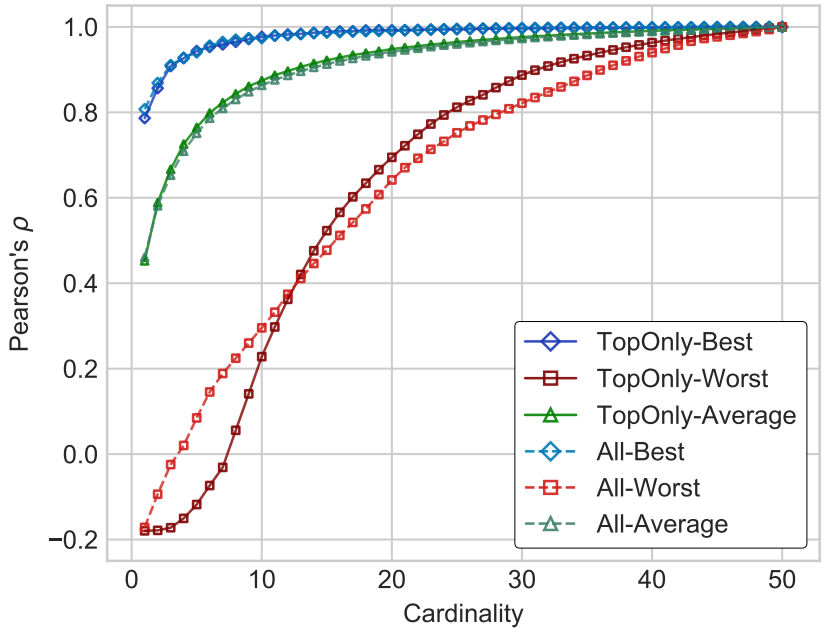
In questa sezione vengono descritti i nuovi esperimenti svolti utilizzando *NewBestSub*, i quali mirano ad estendere gli studi svolti in precedenza. Il primo di tali esperimenti consiste nell'analizzare mediante *NewBestSub* approssimazioni dei dataset indicati nella tabella 8.1 costruite mediante la strategia di selezione a priori dei topic (ossia, prima che un valutatore umano assegni giudizi di relevance ai documenti) sviluppata da Soboroff et al. [34] e nell'analisi dei risultati ottenuti analizzando i punteggi di hubness per una di tali approssimazioni, secondo la metodologia proposta da Mizzaro et al. [25].

Il secondo esperimento consiste nel verificare se i sottoinsiemi *Best/Worst* individuati grazie a tali tecniche sono sottoinsiemi *Best/Worst* anche per i dataset originali. Poiché l'analisi dei punteggi di hubness viene svolta su una delle approssimazioni costruite con la strategia sviluppata da Soboroff et al. [34], la quale è completamente a priori, anche tale tipologia di analisi va considerata come a priori. Gli esperimenti, in particolare, mirano a fornire una metodologia *pratica* per selezionare sottoinsiemi *Best/Worst* di topic senza affidarsi a valutatori umani.

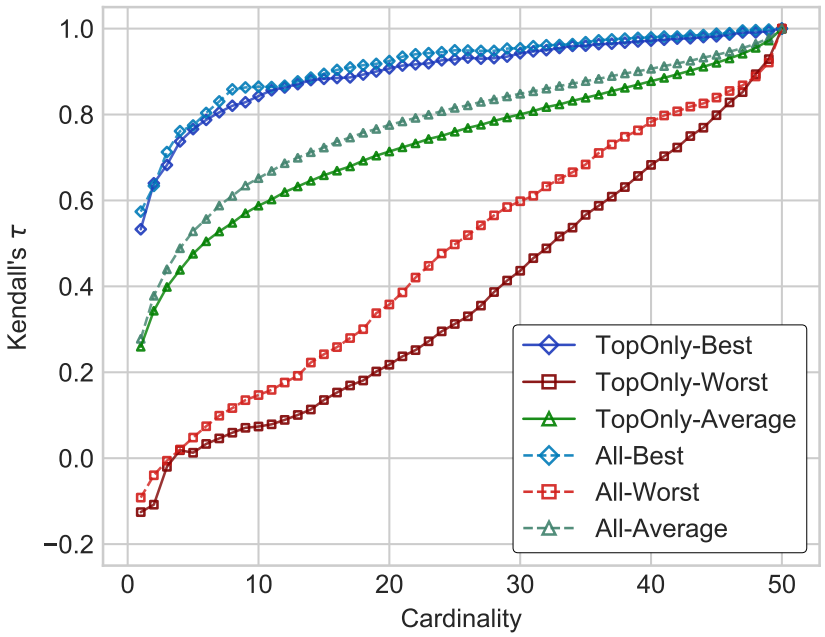
8.4.1 Strategia di Selezione A Priori dei Topic

Per quanto riguarda il primo esperimento, esso si divide in due fasi. Inizialmente, viene costruita una versione approssimata di alcune delle matrici topic per sistema che caratterizzano i dataset della tabella 8.1, utilizzando la strategia di selezione a priori dei topic sviluppata Soboroff et al. [34]. Successivamente, tali matrici approssimate vengono analizzate mediante *NewBestSub*, il quale produce risultati analoghi a quelli descritti nella sezione 8.3, ossia il confronto tra valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti, l'analisi della stabilità dei sottoinsiemi *Best/Worst* ottenuti e l'analisi della stabilità generalizzata ai migliori 10 sottoinsiemi *Best/Worst*.

Al fine di costruire una matrice topic per sistema per la valutazione dell'efficacia dei sistemi di IR, dei valutatori umani devono indicare un giudizio di relevance per ogni documento recuperato da ciascun sistema presente nel pool della collezione di test utilizzata. Tali giudizi di relevance vengono scritti in appositi file chiamati *qrels*,



(a) Valori di correlazione calcolati mediante ρ.



(b) Valori di correlazione calcolati mediante τ.

Figura 8.11: Confronto tra i valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti per i dataset AH99-AP@20/AH99-AP@20-Top96.

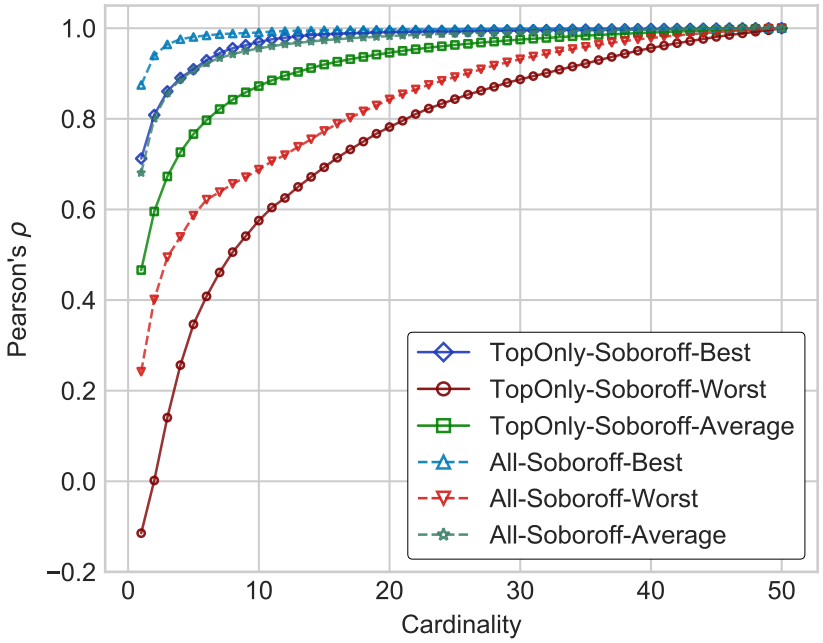
#	Acronimo	Dataset Originale	Num. Topic	Num. Sistemi
1	AH99-Soboroff	AH99	50	129
2	AH99-Top96-Soboroff	AH99-Top96	50	96
3	TB06-Soboroff	TB06	149	61
4	TB06-Top49-Soboroff	TB06-Top49	149	49
5	R04-Soboroff	R04	249	110
6	R04-Top82-Soboroff	R04-Top82	249	82

Tabella 8.7: Collezioni di test approssimate con la strategia di Soboroff et al. [34].

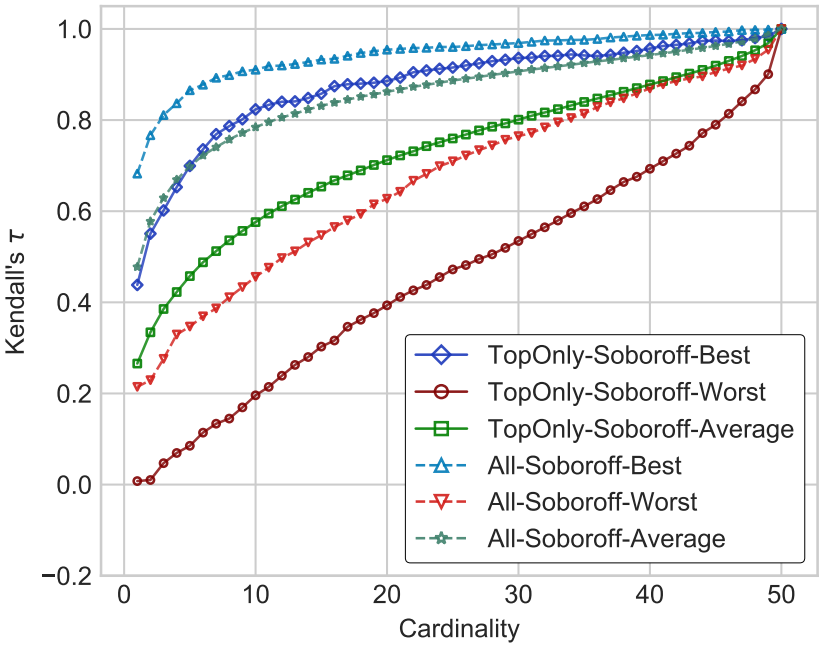
come descritto nella sezione 2.1. La strategia di selezione a priori dei topic sviluppata da Soboroff et al. [34] mira a costruire dei cosiddetti *pseudo-qrels* in un modo completamente automatico, senza l'intervento di alcun valutatore umano, selezionando casualmente documenti dal pool ed indicandoli come pertinenti. Successivamente, a partire da essi viene costruita una matrice topic per sistema analoga a quelle descritte in precedenza (Figura 2.4). Tale strategia risulta avere un'efficacia piuttosto limitata, poiché l'efficacia dei sistemi calcolata sulla base delle matrici approssimate ha una correlazione con quella originale pari solamente a 0.5, nei casi migliori. Ad ogni modo, tale strategia viene effettivamente utilizzata per approssimare alcuni dei dataset indicati nella tabella 8.1, i quali vengono riportati nella tabella 8.7.

Le considerazioni che si possono trarre dai risultati ottenuti da *NewBestSub* sui dataset approssimati sono analoghe a quelle riguardanti i dataset originali. In particolare, nelle figure 8.12 sono visibili gli esiti delle analisi svolte includendo tutti i sistemi o il 75% di quelli più efficaci per l'approssimazione del dataset AH99-Top96, ossia AH99-Top96-Soboroff. Anche per i dataset approssimati si nota come includendo tutti i sistemi si ottengano sottoinsiemi *Best* ed *Average* migliori, mentre nel secondo caso si ottengono sottoinsiemi *Worst* migliori.

Anche per quanto riguarda i risultati di stabilità (Tabella 8.8), i fenomeni sono analoghi a quelli che si verificano per i dataset originali (Tabella 8.5). In altre parole, i risultati sono simili per dataset diversi ed i sottoinsiemi *Worst* sono più stabili di quelli *Best*. Anche per i risultati di stabilità generalizzati ai migliori 10 sottoinsiemi *Best*/*Worst* (Tabella 8.9), si ottiene una situazione analoga a quella originale (Tabella 8.6), dove i sottoinsiemi *Worst* risultano decisamente più stabili di quelli *Best*.



(a) Valori di correlazione calcolati mediante ρ.



(b) Valori di correlazione calcolati mediante τ.

Figura 8.12: Confronto tra i valori di correlazione ottenuti utilizzando il 75% dei sistemi più efficaci ed utilizzandoli tutti per i dataset AH99-Soboroff/AH99-Top96-Soboroff.

#	Dataset	ρ di Pearson		τ di Kendall	
		Best Set	*Worst* Set	*Best* Set	*Worst* Set
1	AH99-Soboroff	.90	.95	.74	.93
2	AH99-Top96-Soboroff	.91	.97	.85	.93
3	TB06-Soboroff	.90	.98	.88	.95
4	TB06-Top49-Soboroff	.89	.98	.88	.93
5	R04-Soboroff	.94	.97	.92	.96
6	R04-Top82-Soboroff	.94	.97	.91	.96
	Media	.91	.97	.87	.94

Tabella 8.8: Valori di stabilità per i sottoinsiemi *Best/Worst* calcolati utilizzando la formula definita da Guiver et al. [20], per le approssimazioni delle collezioni originali.

#	Dataset	Migliori 10 ***Best* Set**						Migliori 10 ***Worst* Set**					
		Cardinalità						Cardinalità					
		5	10	20	30	40	45	5	10	20	30	40	45
1	AH99-Soboroff	.67	.80	.77	.89	.69	.64	.62	.83	.84	.80	.87	.76
2	AH99-Top96-Soboroff	.60	.81	.61	.88	.80	.64	.60	.86	.92	.90	.86	.76
3	TB06-Soboroff	.60	.68	.87	.84	.95	.94	.80	.84	.92	.92	.94	.92
4	TB06-Top49-Soboroff	.53	.78	.90	.89	.88	.95	.67	.82	.93	.93	.96	.95
5	R04-Soboroff	.73	.87	.94	.94	.97	.93	.73	.83	.92	.94	.94	.95
6	R04-Top82-Soboroff	.78	.90	.92	.91	.83	.97	.76	.85	.92	.96	.96	.94
	Media	.65	.80	.83	.90	.85	.85	.70	.84	.90	.91	.92	.88

Tabella 8.9: Valori di stabilità per i migliori 10 sottoinsiemi *Best*/*Worst* individuati a partire dalle approssimazioni delle collezioni originali, con valori di correlazione calcolati mediante τ.

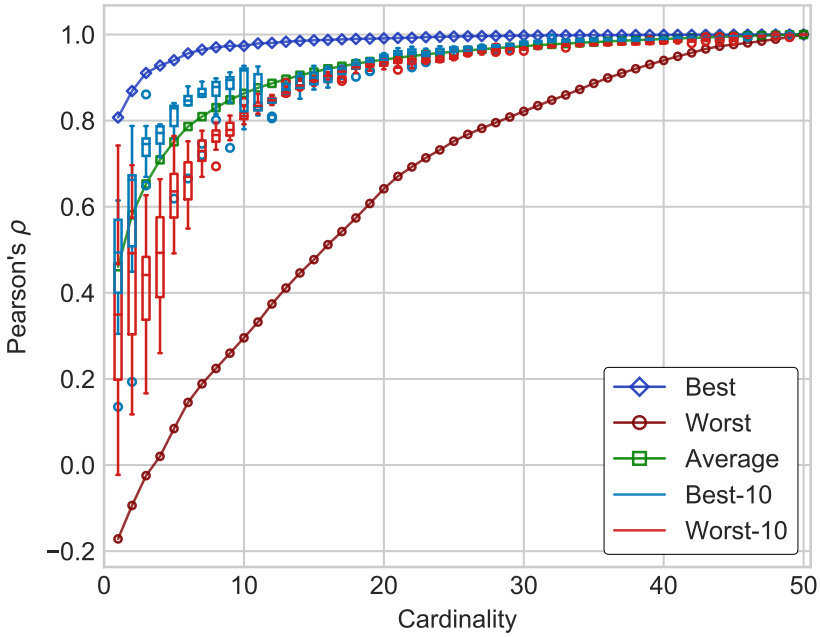
Avendo tale dataset approssimato, il secondo esperimento mira a verificare se i sottoinsiemi *Best*/*Worst* ottenuti analizzando AH99-Top96-Soboroff mediante *NewBestSub* risultano sufficientemente generali per essere sottoinsiemi *Best*/*Worst* anche per AH99-Top96. Al fine di verificare tale ipotesi, vengono selezionati da AH99-Top96-Soboroff i migliori 10 sottoinsiemi *Best*/*Worst*, per ciascuna cardinalità c. Successivamente, viene analizzata la correlazione tra tali sottoinsiemi e quelli originariamente individuati da *NewBestSub* per AH99-Top96, per ciascuna cardinalità.

Le figure 8.13 confrontano i valori di correlazione ottenuti per i sottoinsiemi *Best*/*Worst*/*Average* individuati da *NewBestSub* per AH99-Top96 e quelli ottenuti utilizzando i migliori 10 sottoinsiemi *Best*/*Worst* individuati analizzando AH99-Top96-Soboroff su AH99-Top96. In altre parole, i valori di correlazione dei sottoinsiemi individuati analizzando l'approssimazione del dataset ed utilizzati su quello originale. Osservando tali figure si può notare come il risultato della valutazione dei migliori 10 sottoinsiemi *Best*/*Worst* venga rappresentato, per ogni cardinalità c, mediante boxplot. Tali boxplot mostrano come i sottoinsiemi così individuati risultino generali, ossia come pressoché tutti quelli *Best* (*Worst*) abbiano valori di correlazione al di sopra (al di sotto) di quelli dei sottoinsiemi *Average*. Tale risultato permette di ottenere una strategia a priori di selezione dei topic *pratica* ed *efficace*. Se un ricercatore desidera svolgere esperimenti su sottoinsiemi di topic "buoni", può semplicemente eseguire *NewBestSub* sull'approssimazione del dataset scelto costruita con la tecnica di Soboroff et al. [34] ed utilizzare uno dei 10 migliori sottoinsiemi *Best*/*Worst* individuati per la cardinalità scelta.

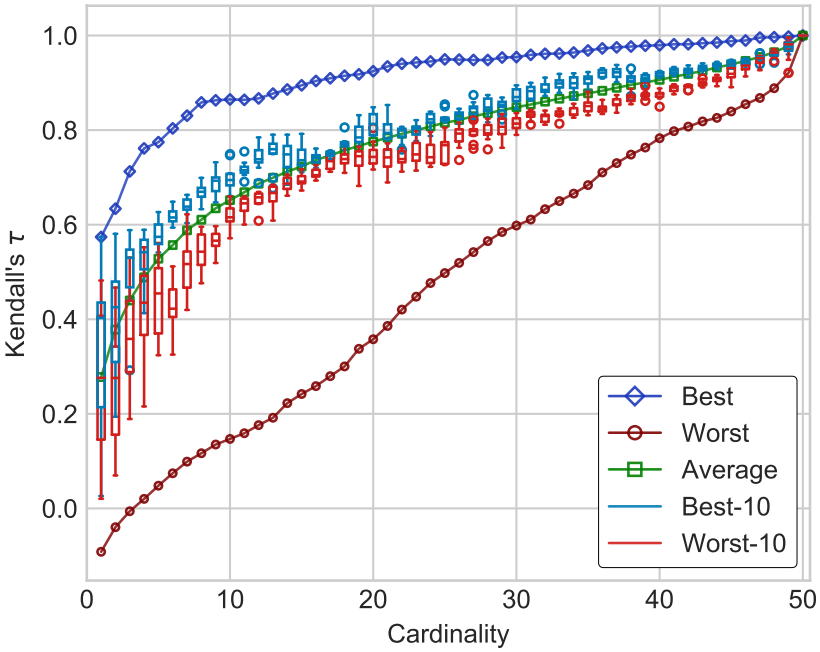
Per quanto riguarda i dataset rimanenti tra quelli indicati nella tabella 8.7, ossia TB06-Top49 (Figura 8.14) e R04-Top82 (Figura 8.15), essi risultano caratterizzati da un numero elevato di topic e si ha una situazione analoga al caso di AH99-Top96 per la maggior parte delle cardinalità, benché in alcuni casi i sottoinsiemi *Best* abbiano valori di correlazioni più bassi dei sottoinsiemi *Average* del dataset originali. La differenza rispetto a tale caso, tuttavia, riguarda i sottoinsiemi *Worst*, poiché nell'ambito di tali sottoinsiemi si ottengono notevoli miglioramenti. Quelli individuati a partire dalle approssimazioni, infatti, risultano caratterizzati da valori di correlazione decisamente più bassi di quelli dei sottoinsiemi *Average*.

8.4.2 Analisi dei Punteggi di *Hubness*

L'analisi dei punteggi di hubness è una metodologia proposta da Mizzaro et al. [25], la quale viene utilizzata da Robertson [29] come strategia per la selezione dei

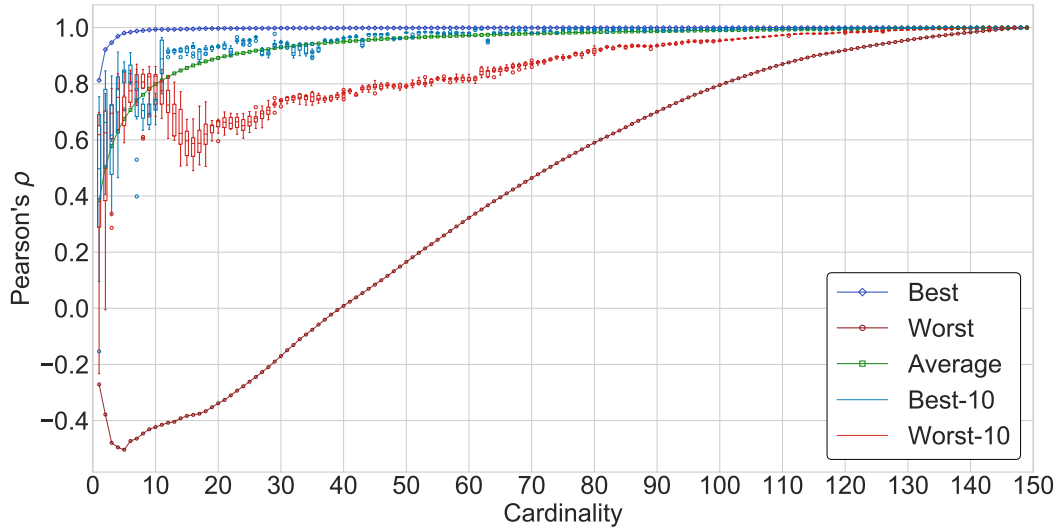


(a) Valori di correlazione calcolati con ρ.

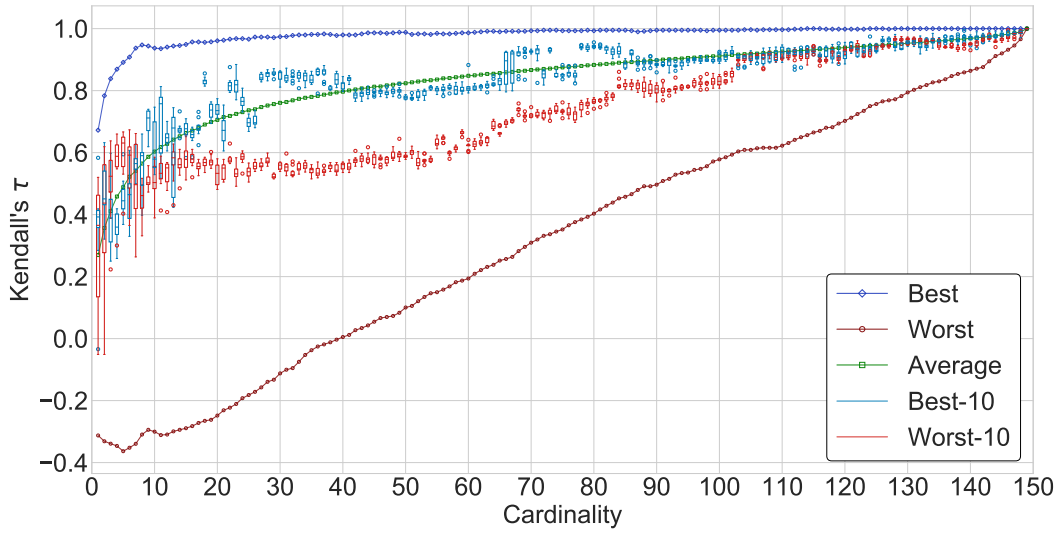


(b) Valori di correlazione calcolati con τ.

Figura 8.13: Confronto tra i valori di correlazione ottenuti analizzando AH99-Top96 mediante *NewBestSub* e quelli ottenuti analizzando AH99-Top96-Soboroff mediante *NewBestSub* ed utilizzando i migliori 10 sottoinsiemi *Best*/*Worst* individuati a partire da tale approssimazione su AH99-Top96.

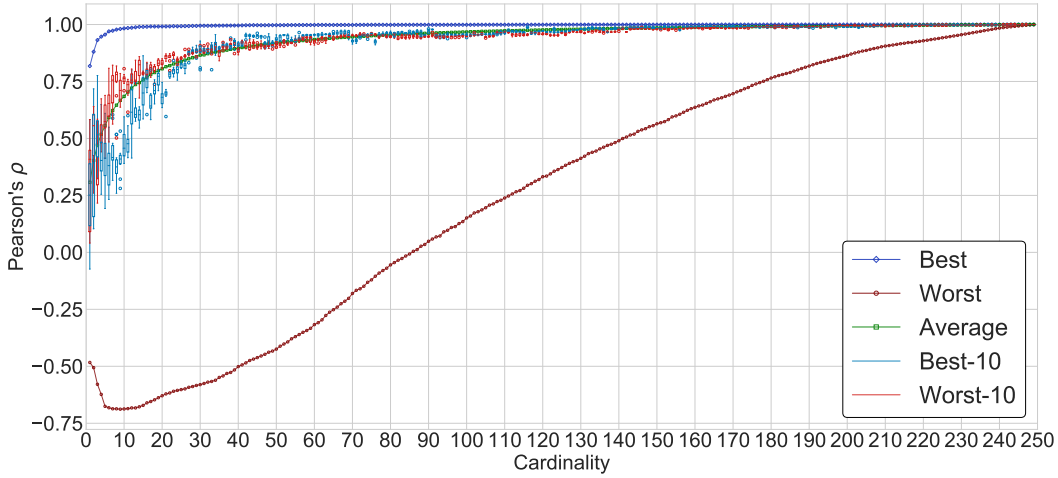


(a) Valori di correlazione calcolati mediante ρ.

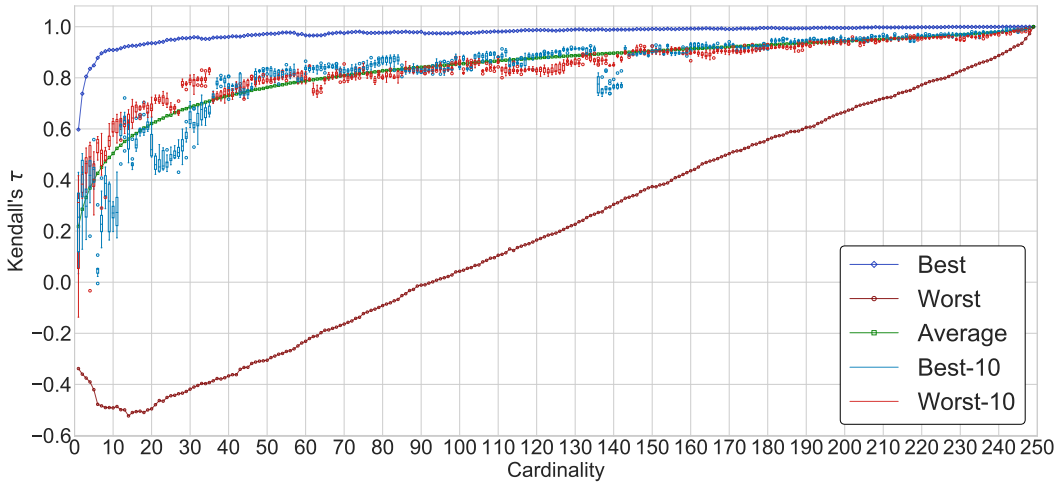


(b) Valori di correlazione calcolati mediante τ.

Figura 8.14: Confronto tra i valori di correlazione ottenuti analizzando TB06-Top49 mediante *NewBestSub* e quelli ottenuti analizzando TB06-Top49-Soboroff mediante *NewBestSub* ed utilizzando i migliori 10 sottoinsiemi *Best*/*Worst* individuati a partire da tale approssimazione su TB06-Top49.



(a) Valori di correlazione calcolati mediante ρ.



(b) Valori di correlazione calcolati mediante τ.

Figura 8.15: Confronto tra i valori di correlazione ottenuti analizzando R04-Top82 mediante *NewBestSub* e quelli ottenuti analizzando R04-Top82-Soboroff mediante *NewBestSub* ed utilizzando i migliori 10 sottoinsiemi *Best*/*Worst* individuati a partire da tale approssimazione su R04-Top82.

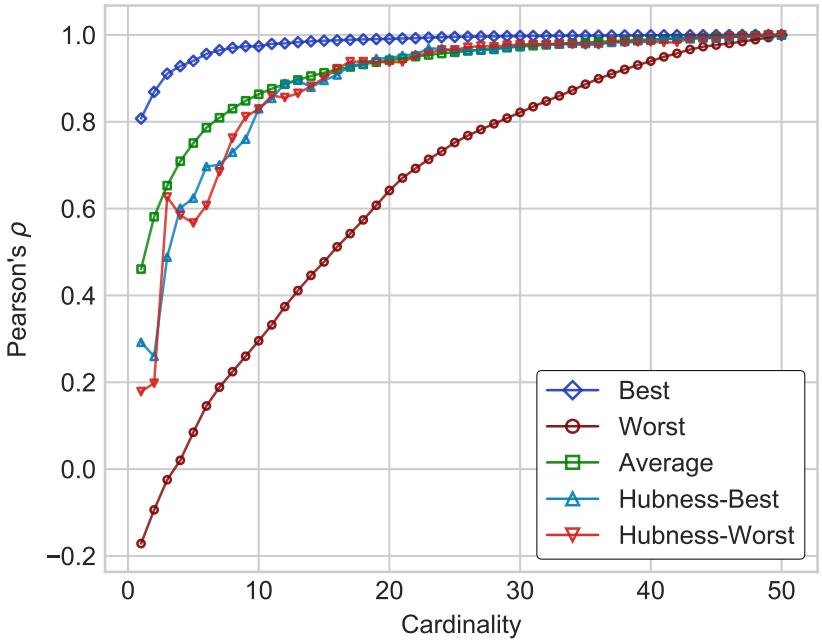
topic poiché permette di calcolare, per ciascuno dei topic stessi, un *punteggio di hubness*, il quale rappresenta l'abilità del topic correntemente analizzato nell'identificare sistemi efficaci. Tale analisi viene svolta su una delle approssimazioni indicate nella tabella 8.7, ossia AH99-Top96-Soboroff.

In particolare, avendo il punteggio di hubness calcolato per ciascuno dei suoi topic, il secondo esperimento mira a verificare se i sottoinsiemi *Best*/*Worst* composti utilizzando topic con valori alti/bassi di hubness sono sufficientemente generali per essere sottoinsiemi *Best*/*Worst* per AH99-Top96. Per verificare tali ipotesi, vengono selezionati sottoinsiemi *Best*/*Worst* di topic di AH99-Top96-Soboroff secondo il loro punteggio di hubness. In particolare, per formare un sottoinsieme *Best*/*Worst* di cardinalità c vengono utilizzati i c migliori topic *Best*/*Worst* ordinati in ordine descrescente/crescente per punteggio di hubness. Infine, viene analizzata la correlazione tra i sottoinsiemi *Best*/*Worst* individuati per AH99-Top96-Soboroff e quelli originariamente individuati da *NewBestSub* per AH99-Top96, per ciascuna cardinalità c.

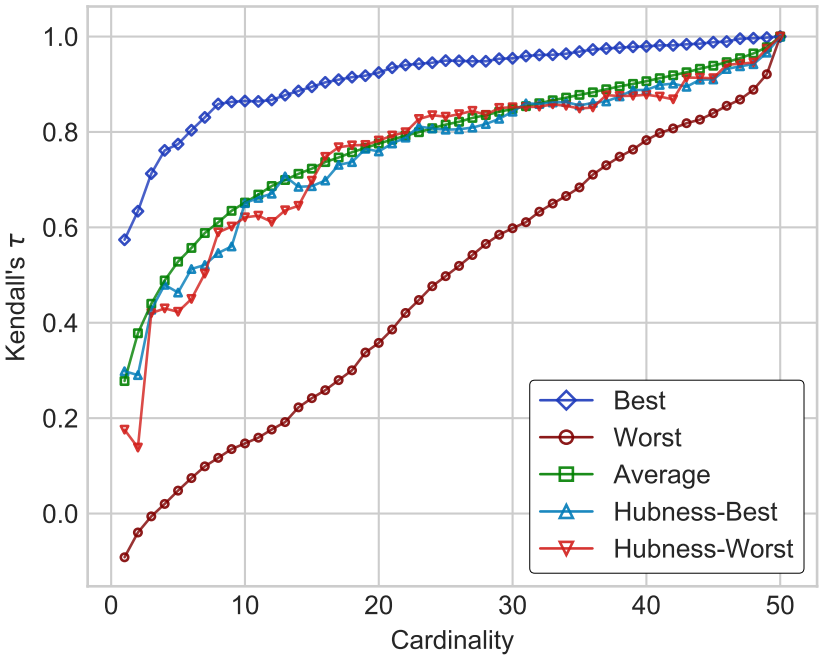
Le figure 8.16 descrivono un confronto analogo a quello descritto nel caso precedente, con la differenza che, per ogni cardinalità c, ad essere confrontati con quelli originali sono i singoli sottoinsiemi *Best*/*Worst* individuati mediante l'analisi dei punteggi di hubness. Tali figure mostrano come i punteggi di hubness calcolati sull'approssimazione del dataset non aiutino a trovare sottoinsiemi di topic "buoni". I valori di correlazione delle serie *Best* e *Worst*, infatti, risultano al di sotto di quelli della serie *Average*, per la maggior parte delle cardinalità. Tale esito negativo può essere causato da diversi fattori, tra i quali:

- l'analisi dei punteggi di hubness può non essere una misura sufficientemente generale da poter essere effettuata sull'approssimazione di un dataset per poi essere utilizzata su quello originale;
- l'approssimazione del dataset può non essere sufficientemente rappresentativa di quello originale. Quella di AH99-Top96 costruita mediante la strategia sviluppata da Soboroff et al. [34], infatti, viene caratterizzata da un valore di correlazione calcolato mediante τ di Kendall pari a 0.5.

Per tali motivi, l'analisi dei punteggi di hubness non può essere sfruttata per ottenere una strategia per permettere ad un qualsiasi ricercatore di effettuare esperimenti su sottoinsiemi "buoni" di topic.



(a) Valori di correlazione calcolati con ρ.



(b) Valori di correlazione calcolati con τ.

Figura 8.16: Confronto tra i valori di correlazione ottenuti analizzando AH99-Top96 mediante *NewBestSub* e quelli ottenuti analizzando i punteggi di hubness di AH99-Top96-Soboroff ed utilizzando i sottoinsiemi *Best*/*Worst* così composti su AH99-Top96.

8.5 Tecnologie

Tutti gli esperimenti descritti nelle sezioni 8.2, 8.3 e 8.4 sono stati svolti a partire dai risultati ottenuti analizzando i dataset indicati nelle tabelle 8.1 e 8.7 mediante *NewBestSub*. Come descritto nella sezione 7.1, tutti i risultati prodotti dal software vengono immagazzinati in file in formato *CSV*. Tale scelta implementativa ha permesso di svolgere gli esperimenti in un modo separato ed indipendente dall'uso del software stesso, utilizzando altre tecnologie più adatte a tale scopo. Le esecuzioni di *NewBestSub*, dunque, possono essere considerate come lo stadio iniziale della fase sperimentale. Dopo la fase di acquisizione dei dati necessari per l'esperimento scelto dai risultati ottenuti utilizzando *NewBestSub* tali dati vengono elaborati e, successivamente, vengono prodotti i risultati finali. Tali risultati, in particolare, possono essere dei grafici o ulteriori file in formato *CSV*.

Mentre l'implementazione di *NewBestSub* è stata costruita utilizzando *Kotlin*, come descritto nella sezione 8.4, per l'esecuzione degli esperimenti descritti in questo capitolo viene utilizzato il linguaggio *Python*[1]. In particolare, viene sfruttata *Anaconda*[2], ossia una distribuzione di tale linguaggio specificamente costruita per le attività di Machine Learning ed analisi di dati. Tale distribuzione contiene centinaia di pacchetti ed, in questa tesi, viene sfruttato un piccolo sottoinsieme di essi.

In particolare, *Pandas*[3] è un pacchetto che fornisce diverse strutture dati e strumenti utili per l'attività di analisi di dati, quali il *Dataframe*[4], struttura utilizzata per rappresentare dati tabellari, ossia determinati casi (righe) dove per ognuno di essi vi sono un numero fissato di osservazioni o misurazioni (colonne). Tale struttura è una scelta piuttosto popolare e viene utilizzata anche in altri ambienti per l'analisi dei dati.

Per la costruzione di tutti i grafici contenuti in questa tesi viene utilizzato il pacchetto *Seaborn*[5], il quale consiste in un'interfaccia di alto livello per la libreria *Matplotlib*[6]. In alcuni casi è risultato necessario svolgere elaborazioni particolari sui grafici costruiti con Seaborn utilizzando direttamente le interfacce di Matplotlib, poiché il primo pacchetto si trova a dover sacrificare le possibilità di personalizzazione a favore della semplicità delle interfacce stesse.

[1] `https://www.python.org/`

[2] `https://www.anaconda.com/what-is-anaconda/`

[3] `https://pandas.pydata.org/`

[4] `https://github.com/mobileink/data.frame/wiki/What-is-a-Data-Frame%3F`

[5] `https://seaborn.pydata.org/`

[6] `https://matplotlib.org/`

Infine, per lo svolgimento generale degli esperimenti, vengono utilizzati i pacchetti *NumPy*[7] e *ScyPy*[8] i quali forniscono strutture dati, interfacce e funzionalità più generali per le attività di analisi di dati e machine learning, quali rappresentazioni efficienti di array multi-dimensionali, implementazioni delle formule di Pearson e Kendall per il calcolo dei valori di correlazione, e quant'altro.

Gli esperimenti vengono svolti utilizzando gli strumenti forniti da tali pacchetti all'interno di alcuni *Jupyter Notebooks*[9]. Un singolo notebook è un'istanza di un'applicazione web che permette di creare e condividere documenti contenenti codice da eseguire, formule, particolari visualizzazioni e semplice testo.

Come si può notare, in questa tesi vi sono stati due momenti distinti anche da un punto di vista puramente tecnologico, i quali consistono nella costruzione e nell'implementazione di *NewBestSub* e nello svolgimento di esperimenti sui risultati ottenuti mediante lo stesso *NewBestSub*.

Nel capitolo 9 vengono presentate le considerazioni finali relative agli esiti di tali attività e i possibili sviluppi futuri relativi all'implementazione di *NewBestSub* ed a nuovi esperimenti da svolgere.

[7] `http://www.numpy.org/`

[8] `https://www.scipy.org/`

[9] `http://jupyter.org/` e `https://try.jupyter.org/`

Capitolo 9

Conclusioni

Lo scopo di questo capitolo consiste nel trarre le conclusioni per quanto riguarda i risultati ottenuti da questa tesi. In particolare, nella sezione 9.1 tali risultati vengono brevemente riepilogati, mentre nella sezione 9.2 vengono descritti i possibili sviluppi futuri, relativi a miglioramenti dell'implementazione di *NewBestSub* e ad ulteriori esperimenti da svolgere.

9.1 Risultati ottenuti

Nel corso di questa tesi è stato possibile progettare ed implementare con successo una nuova versione di *BestSub* (Sezione 2.3), chiamata *NewBestSub* (Capitoli 6 e 7), in grado di superare le limitazioni che caratterizzano lo stesso *BestSub* sfruttando un nuova forma per l'approccio al processo di riduzione del costo della valutazione dei sistemi di IR studiato da Guiver et al. [20] (Sezione 2.2), basata sulla tecnica meta-euristica degli Algoritmi Genetici (Sezione 3.7). L'utilizzo di *NewBestSub*, inoltre, ha reso possibile lo svolgimento di diversi esperimenti insostenibili per *BestSub*.

L'implementazione di *NewBestSub* ha permesso di ottenere diversi vantaggi pratici. Tra quelli più importanti, vi è l'uso di un formato generico di input per i dataset da analizzare e di output per i risultati ottenuti nell'ambito di un esecuzione dello stesso *NewBestSub*. *BestSub*, infatti, risulta strettamente legato alle funzionalità di *Microsoft Excel* per quanto riguarda la produzione dell'output e, dunque, un sistema sprovvisto di tale software non è in grado di eseguire lo stesso *BestSub* in alcun modo. *NewBestSub*, al contrario, utilizza un formato generico (*CSV*) per tali fini, permettendo così di ottenere maggiori portabilità, flessibilità e semplicità d'uso.

Un primo risultato di grande importanza è legato alla maggior efficienza di *NewBestSub* rispetto a quella di *BestSub*. In particolare, tale miglioramento dell'efficienza viene reso possibile grazie all'euristica sulla quale si basa il funzionamento dell'algoritmo genetico *NSGA-II* (Sezione 3.6), ossia quello effettivamente utilizzato per analizzare i sottoinsiemi *Best*/*Worst* di topic (Sezione 3.7). Tale miglioramento consente, dunque, di analizzare dataset caratterizzati da un numero elevato di topic in un tempo sostenibile, laddove l'esecuzione di *BestSub* può richiedere anche diversi mesi. In particolare, un'esecuzione di *NewBestSub* sul più grande dataset disponibile impiega meno tempo per terminare di un'esecuzione di *BestSub* sul più piccolo dataset disponibile (Sezione 8.2). Inoltre, utilizzando *NewBestSub* è possibile riprodurre i risultati ottenuti mediante *BestSub* da Guiver et al. [20] senza alcun sacrificio dal punto di vista dell'efficacia del processo di riduzione del costo della valutazione dei sistemi di IR andando a certificare, in tal modo, l'effettivo funzionamento dello stesso *NewBestSub*.

Un ulteriore risultato di grande importanza viene ottenuto nell'ambito degli esperimenti svolti per generalizzazione e consolidare i risultati ottenuti da Guiver et al. [20] andando ad analizzare quelli ottenuti mediante *NewBestSub* secondo diversi aspetti, per un maggior numero di collezioni di test rispetto a quanto fatto dagli stessi Guiver et al. [20] (Sezione 8.3). In particolare, viene analizzato l'effetto dato dall'inclusione nell'esecuzione dello stesso *NewBestSub* di tutti i sistemi di un dataset o del 75% di quelli più efficaci. Successivamente, viene analizzata la stabilità dei topic contenuti nei sottoinsiemi *Best*/*Worst* individuati e vi è la generalizzazione di tale studio ai migliori 10 sottoinsiemi *Best*/*Worst*, per ogni cardinalità c. Infine, vi l'analisi dell'effetto dato dall'uso di una collezione con una metrica per l'efficacia dei sistemi di IR diversa (*NDCG*) e della tecnica dello shallow pooling. Nell'ambito dei risultati dell'analisi della stabilità dei sottoinsiemi *Best*/*Worst* è possibile concludere che quelli ottenuti da Guiver et al. [20] risultano validi anche con la nuova (e diversa) euristica utilizzata nell'implementazione di *NewBestSub* e, pertanto, è improbabile che siano strettamente dipendenti dall'uso di quella definita da Guiver et al. [20].

Infine, un ulteriore risultato di grande importanza viene ottenuto nell'ambito degli esperimenti svolti al fine di espandere gli studi originali di Guiver et al. [20] (Sezione 8.4). Inizialmente viene utilizzata la strategia di selezione a priori dei topic sviluppata da Soboroff et al. [34] al fine di costruire approssimazioni dei dataset indicati in input. Successivamente, tali approssimazioni vengono analizzate mediante *NewBestSub* e l'analisi dei punteggi di hubness, la quale consiste in una metodologia

proposta da Mizzaro et al. [25], al fine di individuare sottoinsiemi *Best/Worst* di topic con la speranza che tali sottoinsiemi risultino sufficientemente generali da poter essere utilizzati anche per ridurre il costo della valutazione dei dataset originali. Ciò che si ottiene consiste nel fatto che i sottoinsiemi *Best/Worst* individuati eseguendo *NewBestSub* sulle approssimazioni costruite grazie alla strategia sviluppata da Soboroff et al. [34] si rilevano essere effettivamente generali, mentre l'analisi dei punteggi di hubness non si rivela ugualmente efficace.

Ad ogni modo, lo sfruttamento della strategia sviluppata da Soboroff et al. [34] porta alla proposta di una strategia pratica ed efficace di selezione a priori dei topic, la quale permette ad altri ricercatori di svolgere nuovi esperimenti relativi alla riduzione del costo della valutazione dei sistemi di IR evitando di chiamare in causa alcun valutatore umano poiché essi possono ottenere sottoinsiemi "buoni" di topic limitandosi ad eseguire *NewBestSub* sull'approssimazione del dataset scelto (dove tale approssimazione viene costruita mediante la strategia sviluppata daSoboroff et al. [34]) ed ad utilizzare uno dei 10 migliori sottoinsiemi *Best/Worst* individuati in tal modo per la cardinalità da analizzare.

Visti i risultati ottenuti grazie all'uso di *NewBestSub* ed agli esperimenti effettuati nell'ambito di questa tesi, gli obiettivi indicati nel corso capitolo 5 risultano soddisfatti.

NewBestSub è liberamente scaricabile ed utilizzabile al seguente indirizzo:

```
https://github.com/Miccighel/newbestsub
```

9.2 Sviluppi Futuri

Nonostante gli obiettivi indicati nel capitolo 5 risultino soddisfatti, rimane ampio spazio per possibili sviluppi futuri i quali possono essere divisi in due categorie distinte. In particolare, la prima categoria raggruppa miglioramenti all'implementazione di *NewBestSub*, mentre la seconda raggruppa ulteriori esperimenti da svolgere.

Per quanto riguarda la prima categoria, i miglioramenti proposti sono i seguenti:

- svolgere un'attività di fine-tuning dei parametri degli operatori utilizzati manipolare le soluzioni generate durante l'esecuzione dell'algoritmo genetico utilizzato, ossia *NSGA-II* (Sezione 3.6). In particolare, risulta interessante sperimentare l'effetto dato dall'uso di altri operatori logici quali XAND e XOR per svolgere l'attività di *Crossover* (Sezione 3.4);

- studiare i legami esistenti tra la *dimensione della popolazione iniziale* ed il *numero di iterazioni* che *NSGA-II* deve svolgere per ottenere un buon risultato finale e fornire una stima iniziale accurata per la dimensione della popolazione stessa.

- cercare il numero ottimale di esecuzioni di *NewBestSub* delle quali fondere i risultati al fine di trovare ottimi globali migliori;

- modificare l'implementazione di *NewBestSub* affinché sia in grado di individuare il sottoinsieme di topic più generale possibile, ossia quello che massimizza (minimizza) la correlazione con l'insieme originale di topic e l'abilità di essere un sottoinsieme *Best* (*Worst*) per altre collezioni di test.

- studiare un modo per garantire che *NewBestSub* produca dieci (o, più in generale, un numero x) sottoinsiemi *Best*/*Worst* per ciascuna cardinalità della collezione analizzata ed il legame esistente fra tale aspetto ed i parametri degli operatori di *NSGA-II*;

- poiché un'esecuzione di *NewBestSub* può richiedere anche diverse ore per terminare quando vengono analizzate collezioni di test molto grandi, esso dovrebbe essere in grado di sospendere un'esecuzione e salvarne lo stato, in modo da poterla riavviare ricostruendone lo stato stesso ed, in tal modo, terminarla. Ciò può essere utile anche quando l'esecuzione fallisce per via di fattori esterni quali, ad esempio, l'esaurimento dello spazio sul disco all'interno del quale *NewBestSub* viene archiviato. Tuttavia, tale miglioramento richiede anche una modifica all'implementazione di *jMetal* poiché esso, allo stato attuale delle cose, non permette niente di tutto ciò.

Per quanto riguarda la seconda categoria, i consistenti miglioramenti ottenuti nell'efficienza della computazione aprono la strada a diverse ed interessanti possibilità di ricerca come, ad esempio:

- analizzare gli effetti dati da variazioni del numero di topic durante l'analisi di una determinata collezione di test;

- riprodurre ed estendere gli esperimenti di generalizzazione svolti da Guiver et al. [20] utilizzando non solo una singola parte della matrice originale di valori di AP (sia per quanto riguarda la dimensione dei topic che quella dei sistemi), ma svolgendo più iterazioni di tale processo. Tale esperimento, infatti, risulta insostenibile per *BestSub*;

- estendere gli esperimenti di generalizzazione svolti da Guiver et al. [20] analizzando un maggior numero di dataset derivanti dall'applicazione della tecnica dello *shallow pooling* al fine di consolidare le considerazioni tratte nel corso della sottosezione 8.3.4;

- sviluppare ulteriori strategie di selezione a priori dei topic (ossia, senza l'intervento di alcun valutatore umano) integrando altri metodi allo stato dell'arte per costruire approssimazioni dei dataset utilizzati come quelli di Spoerri [36] e Wu et al. [41].

Alla luce dei risultati ottenuti *NewBestSub*, probabilmente, si dimostrerà utile anche nello svolgimento di tali esperimenti e di molti altri a venire.

Ringraziamenti

Una frase che ho sentito ripetere diverse volte è «perché ringrazi qualcuno, dopotutto è solo merito tuo». Ebbene, ciò potrà anche essere vero pensando strettamente agli esami che ho superato oppure a questa tesi che ho scritto, ma in cuor mio son certo che se sono stato in grado di fare tutto ciò è anche grazie a tante persone che mi hanno influenzato positivamente durante questo percorso universitario. Per tale motivo, ringrazio tutti gli amici con i quali ho passato mattine, pomeriggi e notti a studiare, quelli che mi hanno dato consigli preziosi e quelli con cui ho condiviso momenti di gioia e di difficoltà.

Una seconda frase che ho sentito ripetere diverse volte è «perché ringraziare relatore e correlatori, dopotutto hanno solo fatto il loro lavoro». Ebbene, ciò potrà anche essere vero, ma il loro lavoro è stato comunque prezioso per me. Per tale motivo, ringrazio anche il relatore ed i correlatori di questa tesi, ossia il prof. Stefano Mizzaro ed i dott. Kevin Roitero ed Andrea Brunello, per la loro competenza, disponibilità e precisione.

Un ringraziamento particolarmente sentito va alla mia famiglia, la quale ha sempre creduto in me e senza il cui supporto sarei potuto andare ben poco lontano.

Concludo questi ringraziamenti con una citazione la quale, credo, rispecchi perfettamente quello che è stato il mio pensiero durante gli anni universitari, appartenente ad uno scienziato il quale ha rappresentato una grande fonte d'ispirazione per me e gli amici che hanno scelto l'Informatica, fin dai tempi delle scuole superiori.

We can only see a short distance ahead, but
we can see plenty there that needs to be done.

Alan Turing

Michael Soprano *Chiusaforte, 26 Febbraio 2018*

Indice analitico

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Bibliografia

[1] James Allan et al. Overview of the TREC 2007 Million Query Track. In: *Proceedings of the Sixteenth Text REtrieval Conference*. A cura di Ellen M. Vorhees e Lori P. Buckland. Vol. NIST Special Publication: SP 500-274. TREC 2007. National Institute of Standards and Technology, nov. 2007, pp. 1–20.

[2] Ricardo Baeza-Yates e Berthier Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison Wesley, 2011.

[3] David Banks, Paul Over e Nien-Fan Zhang. Blind Men and Elephants: Six Approaches to TREC Data. In: *Information Retrieval* 1.1 (1999), pp. 7–34.

[4] Andrea Berto, Stefano Mizzaro e Stephen Robertson. On Using Fewer Topics in Information Retrieval Evaluations. In: *Proceedings of the 2013 Conference on the Theory of Information Retrieval*. (Copenhagen, Denmark). A cura di Oren Kurland et al. ICTIR '13. ACM, set. 2013, pp. 1093–1100. ISBN: 978-1-4503-2107-5. DOI: 10.1145/2499178.2499184.

[5] Tobias Blickle e Lothar Thiele. A Comparison of Selection Schemes Used in Evolutionary Algorithms. In: *Evolutionary Computation* 4.4 (1997), pp. 361–394. DOI: 10.1162/evco.1996.4.4.361.

[6] Chris Buckley e Ellen M. Vorhees. Evaluating Evaluation Measure Stability. In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. (Athens, Greece). A cura di Emmanuel Yannakoudakis et al. SIGIR '00. New York, NY, USA: ACM, lug. 2000, pp. 33–40. ISBN: 1-58113-226-3. DOI: 10.1145/345508.345543.

[7] Charles Clarke, Nick Craswell e Ian Soboroff. Overview of the TREC 2004 Terabyte Track. In: *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*. A cura di Ellen M. Vorhees e Lori P. Buckland. Vol. NIST Special Publication: SP 500-261. TREC 2004. National Institute of Standards and Technology, nov. 2004, pp. 1–9.

[8] Kevyn Collins-Thompson et al. Overview of the TREC 2014 Web Track. In: *Proceedings of the 23rd Text REtrieval Conference (TREC '14)*. A cura di Ellen M. Vorhees e Angela Ellis. Vol. NIST Special Publication: 500-308. National Institute of Standards and Technology, nov. 2015, pp. 1–21.

[9] Kalyanmoy Deb et al. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. In: *IEEE Transactions on Evolutionary Computation* 6.2 (apr. 2002), pp. 182–197. DOI: 10.1109/4235.996017.

[10] Kalyanmoy Deb e N. Srinivas. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. In: *Evolutionary Computation* 2.3 (1994), pp. 221–248.

[11] A.E. Eiben e J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[12] A.E. Eiben, R. Hinterding e Z. Michalewicz. Parameter Control in Evolutionary Algorithms. In: *Swarm and Evolutionary Computation* 3.2 (lug. 1999), pp. 121–141. DOI: 10.1109/4235.771166.

[13] A.E. Eiben e S.K. Smith. Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms. In: *Swarm and Evolutionary Computation* 1.1 (mar. 2011), pp. 19–31. DOI: 10.1016/j.swevo.2011.02.001.

[14] Roman Elizarov. *Guide to kotlinx.coroutines by example*. 2016. URL: https://github.com/Kotlin/kotlinx.coroutines/blob/master/coroutines-guide.md#concurrent-using-async (visitato il 2017).

[15] Nicola Ferro. Reproducibility Challenges in Information Retrieval Evaluation. In: *Journal of Data and Information Quality* 8.2 (gen. 2017), 8:1–8:4. ISSN: 1936-1955. DOI: 10.1145/3020206.

[16] Nicola Ferro et al. Increasing Reproducibility in IR: Findings From The Dagstuhl Seminar on Reproducibility of Data-Oriented Experiments in E-Science. In: *ACM SIGIR Forum*. Vol. 50. 1. ACM. 2016, pp. 68–82.

[17] Félix-Antoine Fortin e Marc Parizeau. Revisiting the NSGA-II Crowding Distance Computation. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. (Amsterdam, Netherlands). A cura di Christian Blum. GECCO '13. ACM, lug. 2013, pp. 623–630. ISBN: 978-1-4503-1964-5. DOI: 10.1145/2463372.2463456.

[18] Martin Fowler. *UML Distilled: Guida Rapida al linguaggio di Modellazione Standard*. Pearson, giu. 2010.

[19] Erich Gamma et al. *Design Patterns: Elementi per il Riuso di Software ad Oggetti*. Pearson, gen. 2002.

[20] John Guiver, Stefano Mizzaro e Stephen Robertson. A Few Good Topics: Experiments in Topic Set Reduction for Retrieval Evaluation. In: *ACM Transactions on Information Systems (TOIS)* 27.4 (nov. 2009). DOI: 10.1145/1629096.1629099.

[21] David Hawking et al. Overview of the TREC-8 Web Track. In: *The Eighth Text REtrieval Conference (TREC 8)*. A cura di Ellen M. Vorhees e Donna K. Harman. Vol. NIST Special Publication: 500-246. National Institute of Standards and Technology, nov. 1998, pp. 1–18.

[22] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In: *Journal of the ACM* 46.5 (set. 1999), pp. 604–632. ISSN: 0004-5411. DOI: 10.1145/324133.324140. URL: http://doi.acm.org/10.1145/324133.324140.

[23] Christopher D. Manning, Prabhakar Raghavan e Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[24] Stefano Mizzaro. How Many Relevances in Information Retrieval? In: *Interacting With Computers* 10.3 (giu. 1998), pp. 303–320. DOI: 10.1016/S0953-5438(98)00012-5.

[25] Stefano Mizzaro e Stephen Robertson. Hits hits TREC: Exploring IR Evaluation Results with Network Analysis. In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. (Amsterdam, The Netherlands). A cura di Wessel Kraaij et al. New York, NY, USA: ACM, 2007, pp. 479–486. ISBN: 978-1-59593-597-7. DOI: 10.1145/1277741.1277824.

[26] Antonio J. Nebro. *jMetal 5 Documentation*. 2015. URL: https://github.com/jMetal/jMetalDocumentation.

[27] Antonio J. Nebro. *jMetal: a Framework for Multi-Objective Optimization with Metaheuristics*. 2015. URL: https://jmetal.github.io/jMetal/.

[28] Antonio J. Nebro, Juan J. Durillo e Matthieu Vergne. Redesigning the jMetal Multi-Objective Optimization Framework. In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. (Madrid, Spain). A cura di Sara Silva. GECCO '15. ACM, lug. 2015, pp. 1093–1100. ISBN: 1-59593-322-0. DOI: 10.1145/2739482.2768462.

[29] Stephen Robertson. On the Contributions of Topics to System Evaluation. In: *Proceedings of the 33rd European Conference on Advances in Information Retrieval*. (Dublin, Ireland). ECIR 2011. New York, NY, USA: Springer-Verlag, 2011, pp. 129–140. ISBN: 978-3-642-20160-8. DOI: 10.1007/978-3-642-20161-5_14. URL: https://doi.org/10.1007/978-3-642-20161-5_14.

[30] Stephen Robertson. On the Contributions of Topics to System Evaluation. In: *Proceedings of the 33rd annual European Conference on Information Retrieval Research*. (Dublin, Ireland). A cura di P. Clough et al. Vol. 6611. Springer-Verlag, apr. 2011, pp. 129–140. ISBN: 978-3-642-20161-5. DOI: 10.1007/978-3-642-20161-5_14.

[31] Kevin Roitero, Eddy Maddalena e Stefano Mizzaro. Do Easy Topics Predict Effectiveness Better Than Difficult Topics? In: *Advances in Information Retrieval: 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*. A cura di Joemon M. Jose et al. Cham: Springer International Publishing, 2017, pp. 605–611. ISBN: 978-3-319-56608-5. DOI: 10.1007/978-3-319-56608-5_55.

[32] Tetsuya Sakai. Topic Set Size Design. In: *Information Retrieval Journal* 19.3 (giu. 2016), pp. 256–283. DOI: 10.1007/s10791-015-9273-z.

[33] Paolo Serafini. *Ricerca Operativa*. Springer-Verlag, 2009.

[34] Ian Soboroff, Charles Nicholas e Patrick Cahan. Ranking Retrieval Systems Without Relevance Judgments. In: *Proceedings of the 24th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval*. (New Orleans, USA). A cura di Donald H. Kraft et al. SIGIR '01. ACM, set. 2001, pp. 66–73. DOI: 10.1145/383952.383961.

[35] K. Sparck Jones e C. S. Van Rijsbergen. Information Retrieval Test Collections. In: *Journal of Documentation* (1976). DOI: 10.1108/eb026616.

[36] Anselm Spoerri. How The Overlap Between The Search Results of Different Retrieval Systems Correlates With Document Relevance. In: *Proceedings of the American Society for Information Science and Technology* 42.1 (2005), n/a–n/a. ISSN: 1550-8390. DOI: 10.1002/meet.14504201175.

[37] Ellen M. Vorhees. Overview of the TREC 2003 Robust Retrieval Track. In: *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*. A cura di Ellen M. Vorhees e Lori P. Buckland. Vol. NIST Special Publication: SP 500-255. National Institute of Standards and Technology, nov. 2003, pp. 1–9.

[38] Ellen M. Vorhees e Donna Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In: *The Eighth Text REtrieval Conference (TREC 8)*. A cura di Ellen M. Vorhees e Donna K. Harman. Vol. NIST Special Publication: 500-246. National Institute of Standards and Technology, nov. 1998, pp. 1–24.

[39] Ellen M. Vorhees e Donna K. Harman, cur. *Proceedings of the Eighth Text REtrieval Conference (TREC 8)*. Vol. NIST Special Publication: 500-246. National Institute of Standards and Technology, nov. 1998.

[40] William Webber, Alistair Moffat e Justin Zobel. Statistical Power in Retrieval Experimentation. In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. (Napa Valley, California, USA). A cura di James G. Shanahan et al. CIKM '08. New York, NY, USA: ACM, 2008, pp. 571–580. ISBN: 978-1-59593-991-3. DOI: 10.1145/1458082.1458158.

[41] Shengli Wu e Fabio Crestani. Methods for Ranking Information Retrieval Systems Without Relevance Judgments. In: *Proceedings of the 2003 ACM Symposium on Applied Computing*. (Melbourne, Florida). SAC '03. New York, NY, USA: ACM, 2003, pp. 811–816. ISBN: 1-58113-624-2. DOI: 10.1145/952532.952693.

[42] Justin Zobel. How Reliable are the Results of Large-Scale Information Retrieval Experiments? In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. (Melbourne, Australia). A cura di W. J. Croft et al. SIGIR '98. New York, NY, USA: ACM, ago. 1998, pp. 307–314. DOI: 10.1145/290941.291014.