Un classificatore di condizioni non soddisfacibili per migliorare l'efficienza di esecuzione simbolica

Cristian Piacente

Matricola 866020

Relatore: *Prof. Giovanni Denaro* **Co-relatore**: *Prof. Pietro Braione*

Università degli Studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
Corso di laurea in Informatica

Anno Accademico 2022-2023

25 Luglio 2023

Contesto

- Costruzione software di alta qualità $\Rightarrow$ testing
- Generazione automatica test $\Rightarrow$ riduzione costi, esplorazione spazio di input
- Esecuzione simbolica: input $\Longleftrightarrow$ simboli

Esecuzione simbolica dinamica

Combinazione di analisi statica e dinamica per la **generazione automatica di casi di test**

Obiettivo

Massimizzare il numero di casi di test generati in un tempo limitato, evitando il tempo disperso per analizzare program-path non eseguibili

Per raggiungere l'obiettivo:

- adattamento del tool TARDIS
- machine learning
  - viene integrata un'euristica per **classificare** le condizioni simboliche come soddisfacibili o non soddisfacibili

Contenuti

1. Studio dell'esecuzione simbolica
2. Analisi del problema di efficienza
3. Progetto di classificatore
4. Convalida su benchmark di 10 classi di programmi reali

Esecuzione simbolica dinamica (1/2)

Concetti preliminari

Path condition, cammino feasible/infeasible, concolic execution

```
public class MIPC {
  private final int[] a;
  private final int[] b;
  public MIPC(/* 15 int parameters */) {
    a = new int[]{a0, a1, a2, a3, a4};
    b = new int[]{b0, b1, b2, b3, b4, b5, b6, b7, b8, b9};
    for (int elemInB : b) {
      if (elemInB < 0) throw new RuntimeException();
    }
  }
  public boolean target() {
    int k = 0; boolean flag = true;
    for (int i = 0; i < a.length; ++i) {
      if (a[i] > 1000 + (1000 * i)) {
        for (int j = 0; j < b.length; ++j) {
          if (b[j] < -a[i]) { /* infeasible */
            k = 1;
          }
        }
      } else {
        flag = false;
      }
    }
    return (flag && k == 0);
  }
}
```

- Programma esempio particolare

Esecuzione simbolica dinamica (2/2)

```
int k = 0; boolean flag = true;
for (int i = 0; i < a.length; ++i) {
  if (a[i] > 1000 + (1000 * i)) {
    for (int j = 0; j < b.length; ++j) {
      if (b[j] < -a[i]) k = 1;
    }
  }
  else flag = false;
}
return (flag && k == 0);
```

- **Generazione caso di test**

```
MIPC mipc = new MIPC(0, ..., 0);
boolean targetRes = mipc.target();
```

- Calcolo path condition e PC alternative

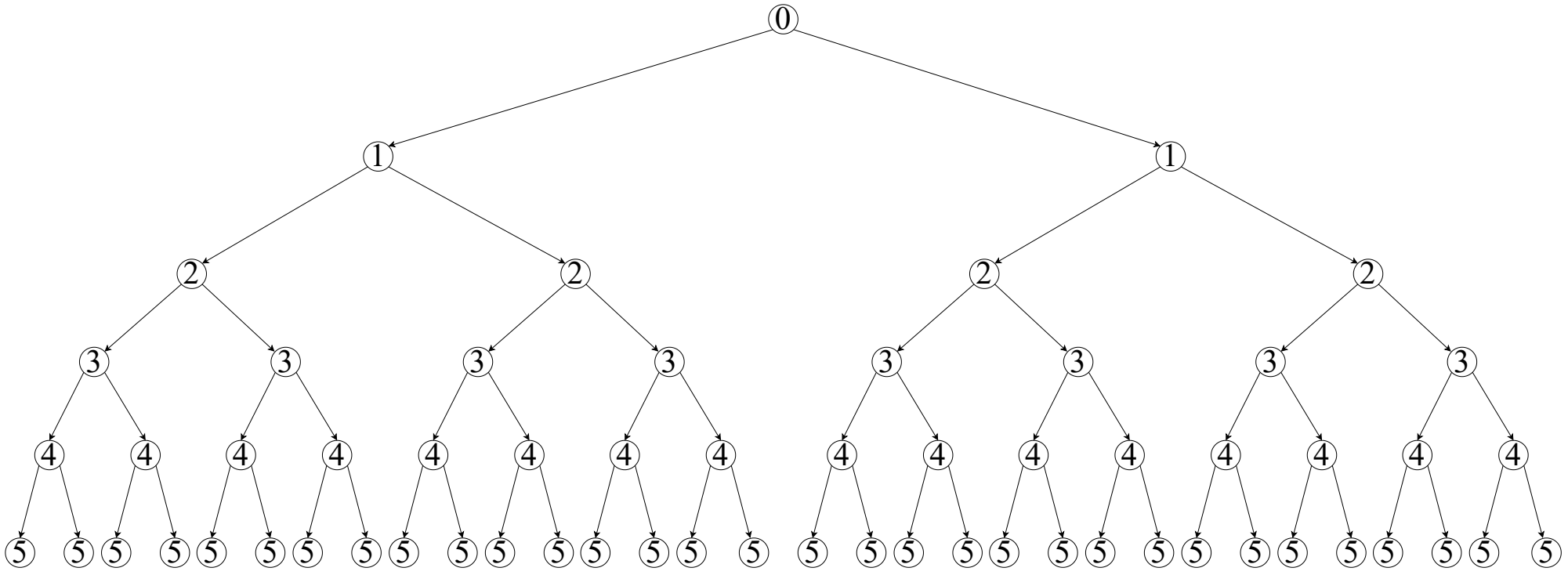
```
A.length ≥ 0 && 0 < A.length && A[0] ≤ 1000 && 1 < A.length &&
 A[1] ≤ 2000 && 2 < A.length && A[2] ≤ 3000 && 3 < A.length &&
 A[3] ≤ 4000 && 4 < A.length && A[4] ≤ 5000 && 5 ≥ A.length
```

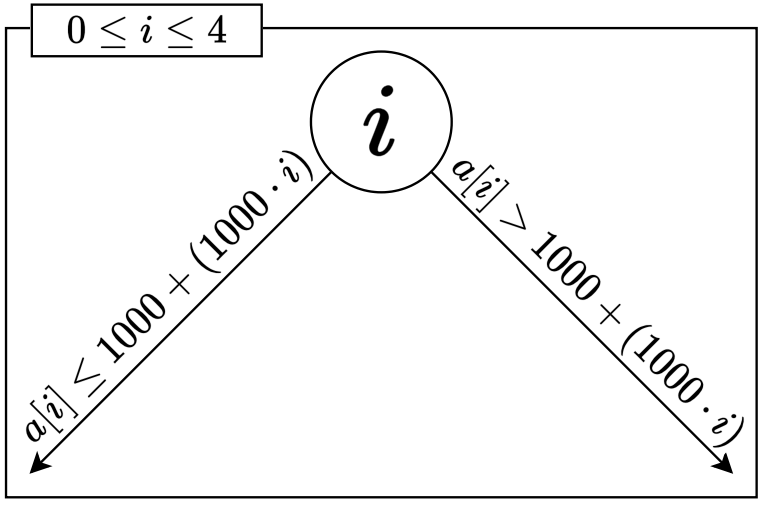
  - `A == null`
  - `A.length ≥ 0 && 0 ≥ A.length`
  - `A.length ≥ 0 && 0 < A.length && A[0] > 1000`
  - `A.length ≥ 0 && 0 < A.length && A[0] ≤ 1000 && 1 ≥ A.length`
  - ⋮
  - `A.length ≥ 0 && 0 < A.length && A[0] ≤ 1000 && 1 < A.length &&`
    `A[1] ≤ 2000 && 2 < A.length && A[2] ≤ 3000 && 3 < A.length &&`
    `A[3] ≤ 4000 && 4 < A.length && A[4] ≤ 5000 && 5 < A.length`

- Scelta path condition da passare ad EvoSuite
  - **Generazione caso di test** oppure timeout

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Cammini feasible





- 2^5 cammini feasible
- Centinaia di path condition non soddisfacibili

Approccio: **classificare** le path condition alternative

Classificatore (1/2)

Separazione in sottoformule

- Chiusura transitiva **relazione di dipendenza**
  - infeasibility core
  - contesto
- Formule concrete e astratte

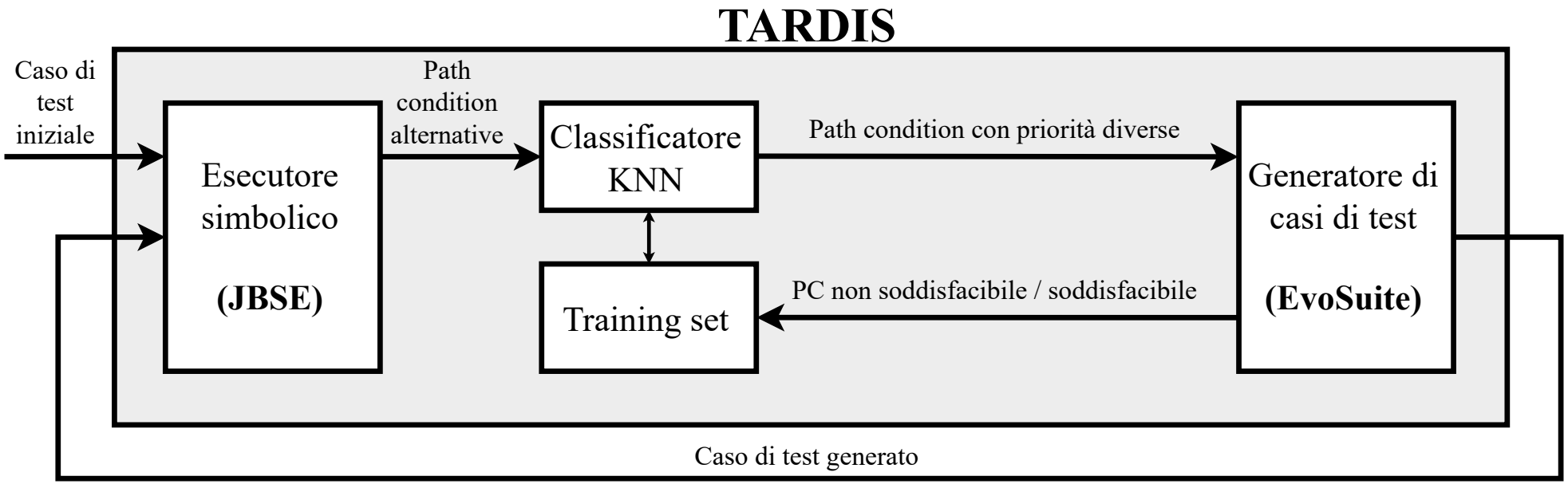
Struttura dati

3 **Bloom Filter**

Classificazione ed estrazione probabilistica

- **KNN** (K = 1)
  - similarità PC alternativa ↔ training items
- Code con diverse probabilità di estrazione
  - PC unknown/feasible: **95%**
  - PC infeasible: 5%

Classificatore (2/2)



Risultati (1/2)

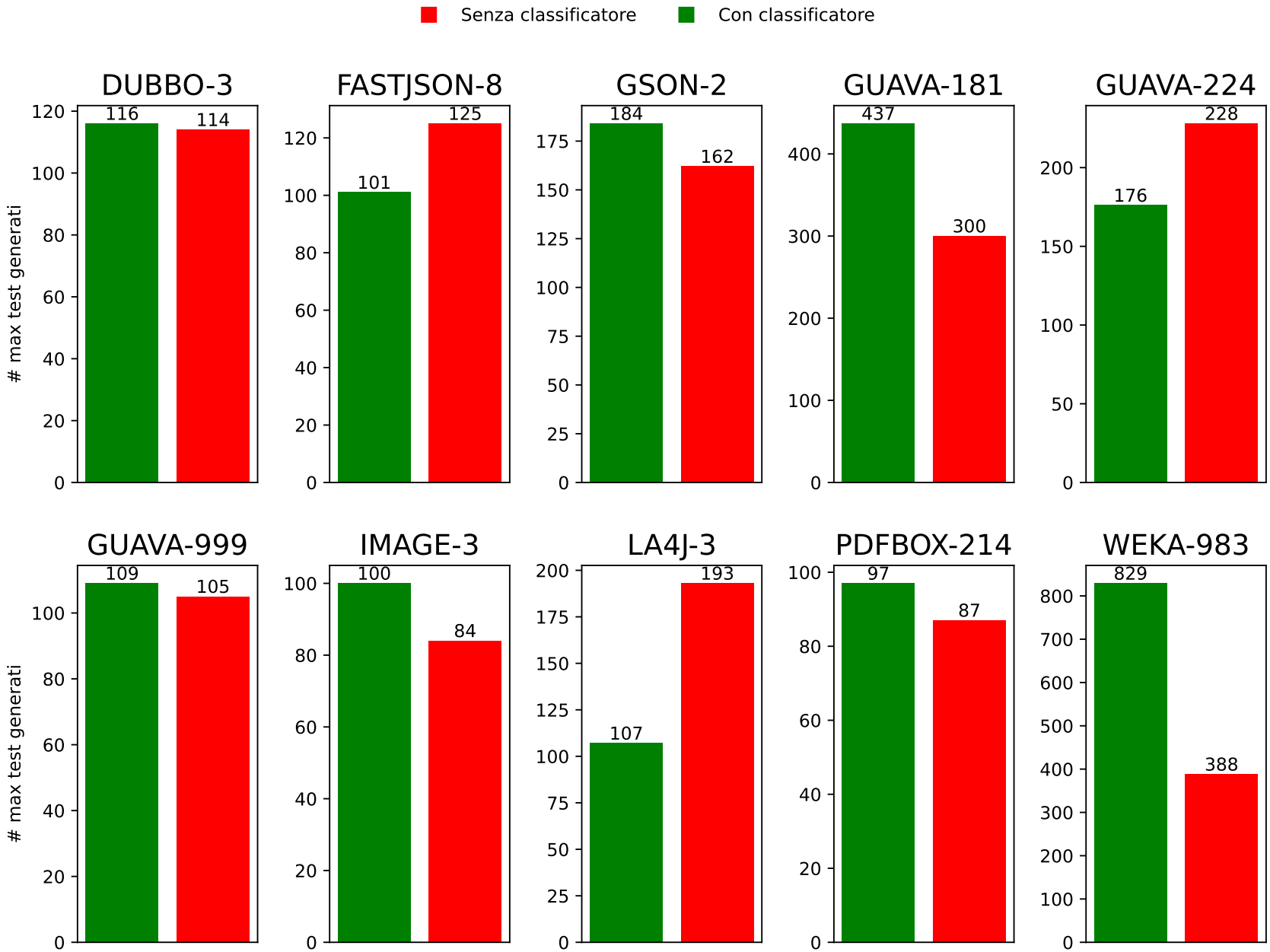
Specifiche tecniche

OS Ubuntu 18.04.4 LTS 64-bit, 48 GB RAM (allocati 16 GB per esecuzione), CPU Intel Xeon(R) Gold 5120 @ 2.20GHz $\times$ 48 cores

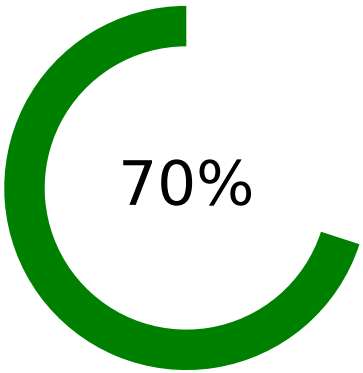
Benchmark

- Parametri di configurazione
  - 5 threads JBSE, 5 threads EvoSuite, 5 path condition alla volta, tempo globale **60 min**, tempo EvoSuite 7.5 minuti per processo
- **10 classi** di progetti reali
  - **6 esecuzioni** per ogni classe
    - 3 con il classificatore attivato
    - 3 senza l'uso del classificatore
  - numero max di test generati

Risultati (2/2)



Conclusioni



Miglioramenti

- Risultato migliore
  - **+441 test**
- Risultato peggiore
  - **-86 test**

Sviluppi futuri

TARDIS attualmente risulta costoso in termini di risorse

- ottimizzazione spazio e tempo classi esistenti
  - evitare che un numero elevato di clausole rallenti l'esecuzione simbolica (e.g. quando si controlla se una PC è ridondante)

Grazie per l'attenzione!

Cristian Piacente

Matricola 866020

Anno Accademico 2022-2023

25 Luglio 2023