Università degli Studi di Salerno

Master degree in Bionformatics and digital health

Natural Computation

Immunocomputing

Authors:
Alessandro Montefusco
Vincenzo Ferrara

Supervisor:
Prof. Angelo Marcelli
Prof. Antonio Della Cioppa

second semester 2019/2020

Contents

Chapter 1

Introduzione

L'*ImmunoComputing* è un paradigma della computazione naturale che trae ispirazione dal funzionamento del sistema immunitario. I sistemi immunitari artificiali sono dei sistemi adattivi che hanno lo scopo di risolvere, principalmente, problemi di *pattern recognition*.
In questi sistemi immunitari artificiali si distingue una famiglia di algoritmi ispirati alla teoria della *selezione clonale*, che spiega i meccanismi attraverso i quali i linfociti B e T migliorano la loro risposta agli antigeni nel tempo. Tale risposta viene detta *maturazione dell'affinità*. In generale, gli algoritmi di selezione clonale sono più comunemente applicati ai domini di ottimizzazione e riconoscimento di modelli.

1.1 Problema: Configurazione di Access Point Wireless

Una piccola città sta progettando come fornire un servizio Internet wireless ai suoi cittadini. È quindi necessario posizionare in diversi punti degli Access Point Wireless per coprire tutti i Client, poiché ciascun punto di accesso ha un raggio di servizio limitato. Per ridurre i costi, una soluzione di progettazione con un numero minimo di punti di accesso e una lunghezza minima dei fili che collegano tali punti di accesso è considerata ottimale. Quindi la problematica da risolvere consiste nel decidere il numero di Access Point (AP) necessari e la loro posizione nella città, in modo da ricoprire quanti più Client possibili.

1.2 Soluzione: CLONALG

Una possibile soluzione al problema è rappresentata da uno degli algoritmi ispirati alla selezione clonale: *CLONALG*. In CLONALG ci sono due popolazioni: una popolazione di antigeni (Ag), rappresentante il problema, e una popolazione di anticorpi (Ab), rappresentante le possibili soluzioni. L'algoritmo cerca, attraverso diverse iterazioni, la popolazione migliore di anticorpi che meglio "combatte" gli antigeni, modificandola ed adattandola nel tempo.
La metafora computazionale è la seguente:

- La popolazione di *antigeni* è rappresentata mediante il dataset fornitoci: l'antigene i-esimo rappresenta la posizione di un Client nello spazio. Ogni Client è rappresentato dalle proprie coordinate x e y.
- La popolazione di *anticorpi* è inizializzata in maniera casuale: l'anticorpo i-esimo rappresenta la posizione dell'AP nella città.

Chapter 2

Analisi dell'algoritmo

2.1 Configurazione degli iperparametri

Per quanto riguarda la sperimentazione e la risoluzione del problema, l'assignment prevedeva di utilizzare un software Java-based: *OptalgToolkit*. Tuttavia, data l'impossibilità di caricare un proprio dataset per la risoluzione del problema proposto, senza andare a modificare od estendere le librerie presenti, si è optato per una variante. Il problema, dunque, è stato analizzato mediante un codice python preso da Git-Hub [1] ed opportunamente modificato ed adattato al problema degli AP.
Data la limitatezza delle risorse e visto il tempo computazionale impiegato dall'algoritmo, in accordo con la classe di "Computazione Naturale", abbiamo deciso di operare su un solo dataset ciascuno e variare un unico iperparametro sul quale fare considerazioni finali. In particolare:

- Gruppi dispari hanno utilizzato "Dataset1" [3], composto al suo interno da 200 Client.
- Gruppi pari hanno utilizzato 'Dataset2"[4], composto al suo interno da 400 Client.

La configurazione di partenza dell'algoritmo sulla quale ci siamo accordati, dopo una serie di prove collettive è la seguente.

```
parameters = {'population_size': 64,
              'random_cells_factor': 0.4,
              'selection_size_factor': 0.5,
              'clone_rate': 100,
              'mutation_rate': 0.2,
              'stop_condition': 50}
```

Figure 2.1: Configurazione base.

Gli iperparametri provati e i loro range di variazione sono:

- *population_size*, rappresenta la popolazione iniziale di anticorpi (AP). I valori provati sono: 16, 32, 64, 96. Valori maggiori causavano un rallentamento eccessivo nella computazione.
- *random_cells_factor*, rappresenta il numero di anticorpi random creati in base alla popolazione di partenza, cioè l'insieme R definito da CLONALG. Il range di variazione scelto è $[0.2; 0.5]$. Anche in questo caso, per valori più alti si ha un rallentamento eccessivo nella computazione. Infatti, il numero di elementi nell'insieme R è dato da:

$$pop\ size \cdot random\ cells\ factor \tag{2.1}$$

- *selection_size_factor*, rappresenta il numero di anticorpi selezionate dalla popolazione di cloni, cioè l'insieme S definito da CLONALG. Il range di variazione scelto è $[0.2; 0.5]$.
- *mutation_rate*, rappresenta la probabilità di mutazione dei cloni nello step di *hypermutation*. Il range di variazione scelto è $[0.2; 0.5]$. Un operatore di mutazione più elevato causerebbe una ricerca casuale nello spazio delle soluzioni e potrebbe influenzare molto le performance dell'algoritmo.

Per quanto riguarda gli altri iperparametri:

- *clone_rate*, rappresenta il tasso di clonazione per ogni elemento della popolazione di partenza. Siccome il problema proposto è un problema di minimizzazione, il tasso di clonazione di un anticorpo è inversamente proporzionale al suo livello di affinità (rappresentante il costo della configurazione degli AP).
- *stop_condition*, rappresenta la condizione di terminazione, che in questo caso è in termini di iterazioni, fissate a 50 poiché si è notato che con un numero maggiore di iterazioni si beccava un minimo che non era tanto differente da un minimo preso nelle prime 50 iterazioni. Inoltre, il valore 50 è legato anche alle risorse (temporali) limitate.

2.2 Configurazione degli AP

Per il calcolo del costo di una configurazione di AP e il calcolo dell'affinità, sono stati stabiliti i seguenti parametri:

- *raggio di un AP* pari a 50;
- *costo di un AP* pari a 100;
- *costo per unità di lunghezza di un cavo* pari a 10;
- *potenza di segnale* di un AP pari a .

2.3 Affinità

Il problema assegnatoci è un problema di ottimizzazione. In particolare, bisogna trovare la configurazione degli AP che abbia il minor costo possibile e che riesca a coprire quanti più Client possibili. La funzione di affinità utilizzata per valutare le performance di una configurazione è la seguente:

$$affinity = k_{cost} \cdot \underbrace{\left(\underbrace{\frac{area_{intersection}}{area_{APs\ in\ range}}}_{[0,1]} + \underbrace{\frac{n_{APs\ in\ range}}{n_{APs}}}_{[0,1]} + \underbrace{\frac{length_{connected\ wires}}{length_{all\ wires}}}_{[0,1]} \right)}_{[0,3 \cdot k_{cost}]} - \underbrace{k_{clients} \cdot n_{clients}}_{[0, k_{clients} \cdot n_{clients}]}$$

Figure 2.2: Funzione calcolo dell'affinity.

I termini k_{cost} e k_{Client} sono rispettivamente una *penalizzazione* e un *bonus*.
Il valore k_{Client} è pari a 0.5 ed è una costante moltiplicativa che va a modulare il numero di Client serviti: più Client vengono coperti, tanto maggiore sarà questo valore.
Mentre k_{cost}, pari a 100, va a modulare il costo della configurazione degli AP.
Tale termine è moltiplicato per tre diversi termini:

- Il primo termine, $\frac{area_intersection}{area_{AP_sinrange}}$, rappresenta di quanto un AP va a coprire la zona di uno o più AP.
- Il secondo termine, $\frac{n_{AP_sinrange}}{n_{AP}}$, rappresenta quanto il centro dell'area coperta da un AP è vicino ad uno o più centri delle aree coperte da altri AP. Tale termine indica effettivamente quanto le aree di diversi AP sono tra di loro sovrapposte.
- Il terzo termine, $\frac{length_{connctedwires}}{length_{allwires}}$, indica il costo di un AP in termine di lunghezza dei cavi utilizzati.

Chapter 3

Workflow della sperimentazione

Come spiegato nei capitoli precedenti, le sperimentazioni sono state condotte grazie al codice python adattato al problema. Il dataset sulla quale sono state condotte le sperimentazioni è "Dataset1" che è composto da 200 Client e l'iperparametro che è stato valutato è *random_cells_factor* (in 2.1). Le performance di CLONALG, delle quattro configurazioni, sono state valutate su 10 seed diversi: per ogni seed, l'algoritmo valuta l'affinity media su 50 iterazioni. I seed sono stati generati tramite un generatore di numeri pseudo-casuali e sono i seguenti: 8915, 1318, 7221, 7540, 664, 6137, 6833, 8471, 9449, 7322. Il numero di trial per ogni configurazione è sicuramente troppo basso; tale scelta è forzata a causa del tempo impiegato per fare una singola computazione. Tale valore è stato poi uniformato per tutti i gruppi in modo da poter confrontare successivamente i risultati. In conclusione, le run effettuate sono per un totale di 40.

3.1 Analisi statistica

I dati ottenuti, sono stati analizzati mediante la piattaforma web **STAC** [2],che permette di fare diversi tipi di test statistici, sia parametrici che non parametrici. Non conoscendo la distribuzione dei nostri campioni, sono stati utilizzati due diversi test non parametrici con un livello di significatività α di 0.05: Friedman-Aligned-Ranks e Quade.Tali test permettono di fare una analisi post-hoc per determinare se l'ipotesi nulla H_0 [1] possa essere rigettata oppure no. L'analisi post-hoc è stata condotta tramite il test "Bonferroni-Dunn".

[1]Ipotesi nulla (H0): la media dei risultati di due o più algoritmi è la stessa.

Chapter 4

Analisi dei risultati

4.1 Iperparametro random_cells_factor

Come si evince dalla figura 4.1, il minimo valore dell'affinità si ha con un valore di *random_cells_factor* pari a 0.5 e un numero di Client coperti di 153/200.

random_cells_factor				
Configurazione	Random_cells_factor	Mean_affinity	Global_cost	Avg_clients
1	0.2	20,95338	66643,59766	151,548
2	0.3	19,55236	66899,10227	154,858
3	0.4	19,19241	66797,66729	153,344
4	0.5	19,08677	66797,60314	153,154

Figure 4.1: Valori medi per le 4 configurazioni su 10 seeds.

Dall'analisi statistica condotta sulle quattro distribuzioni si evince che:

- Secondo Friedman-Aligned-Ranking la configurazione migliore è la quarta e l'ipotesi nulla viene rigettata, ciò significa che le distribuzioni delle quattro configurazioni sono diverse. Analizzando, invece, l'analisi post-hoc di Bonferroni-Dunn, si può notare che le distribuzioni relative alle configurazioni 2, 3 e 4 sono statisticamente uguali, mentre le distribuzioni 1 e 4 sono significativamente diverse (ipotesi nulla rigettata). Quindi, la miglior configurazione da tenere in considerazione è la numero 4.

Statistic	p-value	Result
12.68411	0.00537	H0 is rejected

Ranking

Rank	Algorithm
14.20000	Conf_4
15.50000	Conf_3
19.70000	Conf_2
32.60000	Conf_1

Post-hoc (Using Conf_4 as control method)

Comparison	Statistic	Adjusted p-value	Result
Conf_4 vs Conf_1	3.51942	0.00130	H0 is rejected
Conf_4 vs Conf_2	1.05200	0.87840	H0 is accepted
Conf_4 vs Conf_3	0.24865	1.00000	H0 is accepted

Figure 4.2: Analisi STAC: Friedman / Bonferroni-Dunn

- Secondo Quade la configurazione migliore è la terza e l'ipotesi nulla viene rigettata, ciò significa che le distribuzioni delle quattro configurazioni sono diverse. Analizzando, invece, l'analisi post-hoc di Bonferroni-Dunn, si può notare che le quattro distribuzioni sono statisticamente uguali.

Statistic	p-value	Result
5.08422	0.00641	H0 is rejected

Ranking

Rank	Algorithm
1.96364	Conf_3
2.00000	Conf_4
2.23636	Conf_2
3.80000	Conf_1

Post-hoc (Using Conf_3 as control method)

Comparison	Statistic	Adjusted p-value	Result
Conf_4 vs Conf_1	2.25642	0.07213	H0 is accepted
Conf_4 vs Conf_2	0.29630	1.00000	H0 is accepted
Conf_4 vs Conf_3	0.04558	1.00000	H0 is accepted

Figure 4.3: Analisi STAC: QUADE / Bonferroni-Dunn

Dall'analisi statistica possiamo concludere che le performance dell'algoritmo sono influenzate dall'iperparametro analizzato. Infatti, l'affinità media della miglior configurazione, sui 10 trial, è di 19.087, ma il numero di Client serviti e il Global Cost rimangono pressoché invariati rispetto le altre configurazioni.

Si è tenuto in considerazione anche l'andamento della stessa affinità, per ogni configurazione, nel corso delle 50 iterazioni e mediata sulle 10 trial. Come si evince dai grafici riportati in figura 4.4, dopo 20 iterazioni le curve risultano essere pressoché costanti, a meno di qualche perturbazione dovuta alla varianza, visto che raffigurano le affinità medie.

Per tale ragione, come precedentemente accennato, la stop_condition è stata fissata a 50. Sono stati anche esaminati casi con stop_condition con il valore di 100 ma i risultati pervenuti sono stati statisticamente identici a quelli ottenuti con 50.

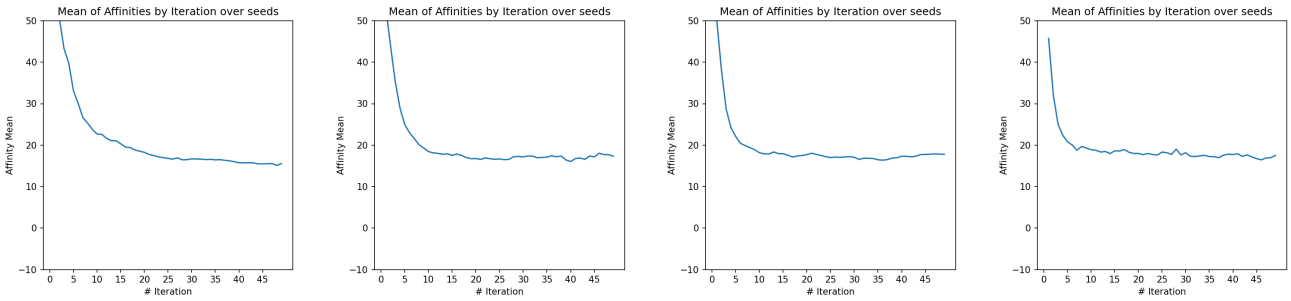

Figure 4.4: Random_cells_factor: 0.2, 0.3, 0.4, 0.5

4.2 Altri iperparametri

Tenendo conto dei risultati ottenuti dai nostri colleghi, sempre analizzando le configurazioni mediante STAC, è stato riscontrato una notevole differenza delle performance. In particolare,

assume un peso considerevole l'iperparametro *population_size*.
Analizzando i dati si è osservato che le migliori performance si ottengono per un numero di AP pari a 64. Con un numero di AP pari a 96 si ha una copertura di Client leggermente maggiore (da 155 a 180, in media) ma si ha un costo globale molto alto (da 66423 a 82932).
Dall'analisi statistica si evince che, osservando le quattro migliori configurazioni, non vi è alcuna differenza (ipotesi nulla accettata). Tuttavia, i test di ipotesi Quade e Friedman-Aligned-Ranking, suggeriscono che la miglior configurazione, in termini di numero di AP, costo globale e Client serviti, è la seguente:

```
parameters = {'population_size': 64,
              'random_cells_factor': 0.5,
              'selection_size_factor': 0.4,
              'clone_rate': 100,
              'mutation_rate': 0.2,
              'stop_condition': 50}
```

Figure 4.5: Configurazione migliore.

Partendo da questa configurazione, abbiamo deciso di approfondire l'analisi dei costi facendo crescere il numero di AP (*population_size*) con passo diverso. In particolare sono stati provati i seguenti valori: 70, 75, 85, 96. Il numero di trial per ogni configurazione è di 10, in modo da poter confrontare i risultati ottenuti.

L'analisi statistica (vedi figure 4.6 e 4.7) ha evidenziato che la miglior configurazione è quella con una *population_size* di 75 AP.

Post-hoc (Using Conf75 as control method)

Comparison	Statistic	Adjusted p-value	Result
Conf75 vs Conf96	4.15062	0.00010	H0 is rejected
Conf75 vs Conf85	2.71608	0.01982	H0 is rejected
Conf75 vs Conf70	1.47280	0.42241	H0 is accepted

Figure 4.6: Test di Aligned-Friedman-Ranking.

Post-hoc (Using Conf75 as control method)

Comparison	Statistic	Adjusted p-value	Result
Conf75 vs Conf96	2.96297	0.00914	H0 is rejected
Conf75 vs Conf85	1.75499	0.23778	H0 is accepted
Conf75 vs Conf70	1.16240	0.73522	H0 is accepted

Figure 4.7: Test di Quade.

In particolare:

- per Friedman, le configurazioni sono statisticamente diverse, tranne quelle con *population_size* di 70 e 75.
- per Quade, le configurazioni sono statisticamente simili, tranne quelle con *population_size* di 75 e 96.

Facendo una analisi anche sui costi globali e sui Client coperti si ottengono i seguenti risultati:

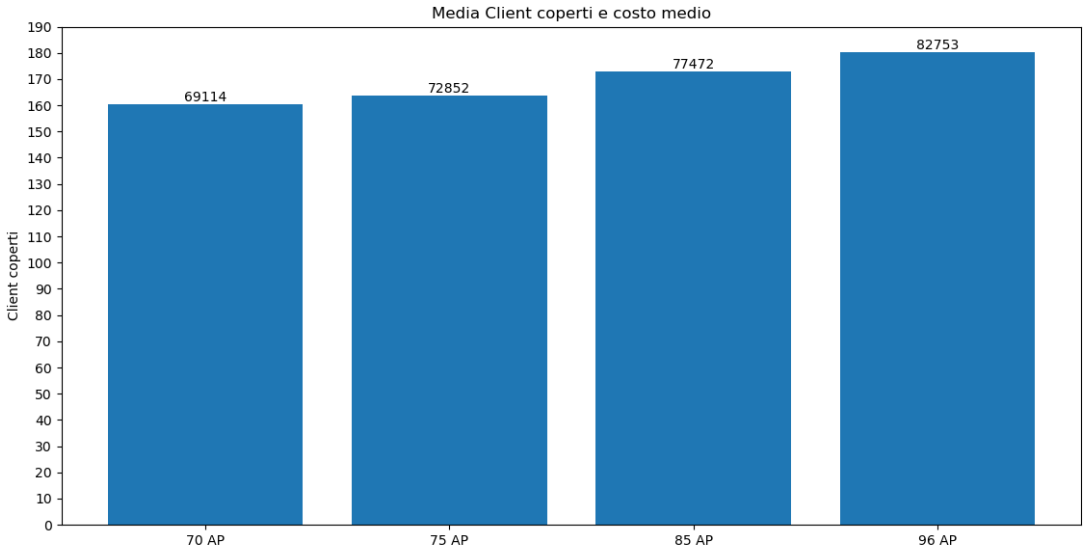

Figure 4.8: Analisi finale dei costi.

Come si evince dalla figura 4.8, all'aumentare del numero di AP aumenta anche il costo complessivo, ma con un numero di Client coperti maggiore. Tuttavia, il notevole incremento del costo non porta alla scelta di un numero di AP elevato dato che l'incremento dei Client coperti non è significativo.

4.3 Conclusioni

Alla luce dell'analisi dei risultati potremmo suggerire di utilizzare un numero di AP pari ad 85, in modo da coprire un numero maggiore di Client (173) con un costo di +11000 rispetto alla configurazione in figura 4.5. Anche l'analisi statistica effettuata suggerisce che queste due configurazioni sono tra loro statisticamente differenti.

```
parameters = {'population_size': 85,
              'random_cells_factor': 0.4,
              'selection_size_factor': 0.4,
              'clone_rate': 100,
              'mutation_rate': 0.2,
              'stop_condition': 50}
```

Figure 4.9: Configurazione suggerita.

Bibliography

[1] Il codice Git-Hub: `https://github.com/christianrfg/clonalg`

[2] Sito Web utilizzato per l'analisi statistica: `http://tec.citius.usc.es/STAC/`

[3] Dataset 1 Wireless Access Point.txt

[4] Dataset 2 Wireless Access Point.txt