UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria
dell'Informazione per Videogame e Realtà Virtuale

Game Design di un Ambiente Competitivo Multi-Criterio Tramite Modelli di Rete e Ottimizzazione Combinatoria

Game Design of a Multi-Criteria Competitive Environment via Network Models and Combinatorial Optimization

Relatore:
Prof. Fabrizio Marinelli

Candidato:
Mario Nicola Petrella

Anno Accademico 2025-2026

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Triennale in Ingegneria dell'Informazione per Videogame e Realtà Virtuale
Via Brecce Bianche – 60131 Ancona (AN), Italy

Sommario

Capitolo 1

Introduzione

1.1 Contesto e Motivazioni

Negli ultimi decenni, l'Ingegneria Informatica e le metodologie didattiche avanzate hanno vissuto una progressiva e proficua convergenza. All'interno delle discipline STEM (Scienza, Tecnologia, Ingegneria e Matematica), lo studio di modelli teorici di ottimizzazione si scontra spesso con barriere cognitive dovute all'elevato grado di astrazione formale. La Ricerca Operativa, la Teoria dei Grafi e l'Ottimizzazione Combinatoria richiedono infatti l'assimilazione di paradigmi algebrici e topologici complessi che, se insegnati con metodologie puramente tradizionali, rischiano di oscurare l'intuizione geometrica e spaziale dei problemi trattati.

In questo scenario, l'impiego dei *Serious Games* – strumenti ludico-educativi progettati non per il puro intrattenimento, ma per scopi formativi espliciti – rappresenta una frontiera metodologica rivoluzionaria. La traduzione di un teorema matematico all'interno di un videogioco permette allo studente di esplorarne empiricamente il significato: la manipolazione interattiva delle variabili genera un feedback visivo immediato, innescando cicli di apprendimento basati sul *problem-solving* esplorativo e promuovendo una comprensione profonda dei fenomeni di causa-effetto.

Molto spesso, la comprensione di algoritmi fondamentali su reti – quali il calcolo del flusso massimo, l'individuazione delle clique o la risoluzione dei complessi problemi di *Network Interdiction* bi-livello – risulta impervia su carta. Lo scopo di questo lavoro è dimostrare come sia possibile decodificare e "mascherare" tali problemi matematici (tecnica dell'*Algorithmic Shrouding*) all'interno di un'ambientazione ludica universale e familiare: una partita di calcio. Il campo da gioco si trasforma rigorosamente in un grafo orientato, in cui gli atleti rappresentano i nodi logici e le traiettorie dei passaggi costituiscono gli archi, le cui portate variano dinamicamente in base a stringenti vincoli spaziali e tattici.

1.2 Obiettivo del Progetto "OptiSoccer"

Il fulcro del presente lavoro di tesi è la progettazione e lo sviluppo software di **OptiSoccer**, un prototipo videoludico realizzato sfruttando il motore tridimensionale *Unity*. L'obiettivo primario di OptiSoccer è fungere da laboratorio algoritmico virtuale

in cui l'utente può sperimentare, testare e validare complesse logiche di Ricerca Operativa travestite da strategie calcistiche.

Il gameplay è ingegnerizzato per riprodurre le fasi tipiche dell'ottimizzazione su reti. In una prima fase decisionale (il *Draft*), il giocatore affronta una variante del Problema dello Zaino (*Knapsack Problem*) per creare la propria formazione, bilanciando un budget finanziario con i requisiti topologici dei giocatori, quali il raggio d'azione (*PassRange*) e la forza fisica (*Attack* e *Defense Power*).

Sul campo di gioco, lo schieramento geometrico dei nodi genera istantaneamente un grafo orientato, in cui il portiere assume il ruolo algebrico di sorgente e il centravanti quello di pozzo. La dinamica centrale implementa un modello computazionale di *Network Interdiction*: le due squadre si affrontano in una fase di *Combat* simultaneo, in cui le probabilità di successo dei tackle riducono o recidono le portate degli archi avversari.

Infine, l'infrastruttura logica di OptiSoccer esegue valutazioni prestazionali in *real-time* tramite l'implementazione in C# di celebri algoritmi su grafi: calcola il flusso offensivo massimo superstite sfruttando l'algoritmo di **Edmonds-Karp** ed analizza l'impermeabilità del blocco difensivo ricercando la **Clique Massima**. Il confronto tra queste due metriche restituisce, in forma di "gol", la prova tangibile della superiorità del modello topologico creato dal giocatore.

1.3 Struttura della Tesi

L'elaborato è organizzato in sei capitoli, strutturati per accompagnare il lettore in un percorso progressivo che parte dall'astrazione teorica, attraversa l'ingegnerizzazione del software e giunge all'analisi empirica dei risultati:

- Il **Capitolo 1** (il presente capitolo) introduce il contesto accademico, il framework pedagogico dei Serious Games per le materie STEM e le finalità progettuali di OptiSoccer.
- Il **Capitolo 2** getta le basi scientifiche del lavoro. Presenta le definizioni formali della Teoria dei Grafi, analizza matematicamente il problema del Flusso Massimo e il teorema Max-Flow Min-Cut, approfondendo le complessità degli algoritmi di Edmonds-Karp e Bron-Kerbosch, per poi formalizzare i modelli di competizione nei *Network Games*.
- Il **Capitolo 3** descrive l'architettura logica e informatica di OptiSoccer sviluppata in Unity. Dettaglia la traduzione del Problema dello Zaino per la gestione del budget di squadra e introduce l'innovativa formulazione matematica dell'*Affinity Multiplier*, la penalità spaziale non lineare basata sul decadimento cubico.
- Il **Capitolo 4** si addentra nel cuore del motore di calcolo del software. Analizza la matematica probabilistica della fase di *Combat* (Interdizione), per poi analiz-

zare criticamente le scelte implementative e di ottimizzazione degli algoritmi di calcolo del flusso e delle clique (es. la discretizzazione numerica e il pruning).

- Il **Capitolo 5** riporta i dati empirici emersi dalle sessioni di collaudo. Valuta l'efficacia di differenti approcci tattici (come il Tiki-Taka o il Contropiede), analizza gli esiti delle formazioni spazialmente estreme (casi limite) e traccia le future prospettive di ricerca sull'automazione IA e sui grafi continui.
- Infine, il **Capitolo 6** riassume i risultati tecnologici e didattici raggiunti, consolidando il valore del progetto all'interno del connubio tra ingegneria informatica e scienze della formazione.

Capitolo 2

Grafi e Problemi Combinatori

In questo capitolo vengono presentati i fondamenti teorici e metodologici che costituiscono l'ossatura scientifica dell'intero elaborato. La trattazione adotta un approccio rigorosamente astratto e formale, delineando i modelli matematici e gli algoritmi di ottimizzazione che governano lo studio delle reti complesse. Nella prima sezione vengono richiamati i concetti cardine della Teoria dei Grafi e delle reti di flusso, formulando matematicamente il problema del Flusso Massimo come modello di Programmazione Lineare. A seguire, si analizza il fondamentale teorema del Max-Flow Min-Cut e si entra nel dettaglio dell'algoritmo di Edmonds-Karp e della teoria delle clique nei grafi non orientati, con particolare riferimento all'algoritmo di Bron-Kerbosch. La seconda sezione è dedicata ai Giochi su Reti e ai modelli bi-livello di Network Interdiction, formalizzando le dinamiche di contrasto e deterioramento delle capacità degli archi in scenari competitivi. Infine, l'ultima sezione esamina la pedagogia dei Serious Games e del Game-Based Learning, analizzando l'efficacia didattica del mascheramento algoritmico per l'apprendimento intuitivo delle discipline scientifiche (STEM).

2.1 Ricerca Operativa e Teoria dei Grafi

La formalizzazione matematica delle relazioni spaziali, della connettività e dello scambio di risorse all'interno di una rete complessa trova il suo strumento d'elezione nella Teoria dei Grafi, una branca fondamentale della Ricerca Operativa. L'astrazione dei sistemi fisici sotto forma di nodi e archi consente di applicare modelli rigorosi per ottimizzare la trasmissione di flussi informativi, logistici o materiali. Questa astrazione è particolarmente potente in quanto indipendente dalla natura fisica della risorsa: che si tratti di bit in una rete telematica, di veicoli in una rete stradale o, nel contesto di questo elaborato, del livello di affinità e l'interazione in campo tra i giocatori, le leggi topologiche che ne governano l'ottimizzazione rimangono invariate.

2.1.1 Formalizzazione del Grafo e Matrice di Adiacenza

Un grafo orientato (o digrafo) è definito formalmente come una coppia ordinata $G = (V, A)$, dove:

- $V = \{v_1, v_2, \ldots, v_n\}$ rappresenta l'insieme finito dei nodi (o vertici). La cardinalità di tale insieme, denotata con $|V| = n$, definisce l'ordine del grafo.
- $A \subseteq V \times V$ rappresenta l'insieme degli archi orientati (o collegamenti). Un arco orientato $a = (u, v) \in A$ indica una connessione unidirezionale avente come origine il nodo u e come destinazione il nodo v. La cardinalità di tale insieme è denotata con $|A| = m$.

Ad ogni arco $(u, v) \in A$ viene associata una capacità pesata non negativa, espressa tramite una funzione di capacità $c : A \to \mathbb{R}^+$. Il valore $c(u, v)$ rappresenta la portata massima teorica, il limite superiore di trasmissione, o la robustezza strutturale del canale di comunicazione tra i nodi u e v.

La struttura topologica e le relazioni di adiacenza del grafo G sono descritte e memorizzate tramite una matrice di adiacenza pesata $W \in \mathbb{R}^{n \times n}$, definita come:

$$W_{i,j} = \begin{cases} c(v_i, v_j) & \text{se } (v_i, v_j) \in A \\ 0 & \text{altrimenti} \end{cases} \tag{2.1}$$

La matrice W risulterà simmetrica se e solo se per ogni coppia di nodi adiacenti esiste una connessione bidirezionale con identica capacità ($c(u, v) = c(v, u)$). Nella stragrande maggioranza delle reti reali, tuttavia, W presenterà valori asimmetrici, riflettendo la direzionalità intrinseca dei flussi fisici.

2.1.2 Formulazione Matematica del Flusso Massimo

Una rete di flusso è un caso particolare di grafo orientato pesato $G = (V, A, c, s, t)$ in cui sono identificati e distinti due nodi speciali aventi ruoli antitetici:

- La **sorgente** $s \in V$, caratterizzata dalla proprietà di generare il flusso all'interno della rete. In molti modelli teorici si assume che s abbia un grado di ingresso nullo.
- Il **pozzo** (o destinazione) $t \in V$, caratterizzato dalla proprietà di assorbire indefinitamente il flusso della rete. Simmetricamente alla sorgente, in alcuni modelli si assume che t abbia un grado di uscita nullo.

Si definisce **flusso** su G una funzione reale $f : A \to \mathbb{R}^+$ che associa a ciascun arco (u, v) un valore scalare $f(u, v)$ soddisfacente rigorosamente i seguenti due vincoli operativi fondamentali:

1. **Vincolo di Capacità:** Il flusso instradato su ciascun arco non può in alcun caso eccedere la portata massima del canale ed è strettamente non negativo. L'arco si comporta come una tubatura fisica:

$$0 \leq f(u, v) \leq c(u, v) \quad \forall (u, v) \in A \tag{2.2}$$

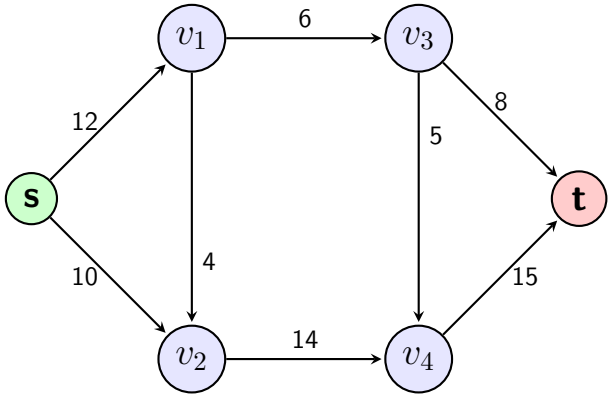

Figura 2.1: Esempio di una rete di flusso orientata. I pesi sugli archi indicano le capacità $c(u, v)$. Il nodo verde s rappresenta la sorgente, il nodo rosso t il pozzo.

2. **Vincolo di Conservazione del Flusso (Equilibrio dei Nodi):** Per ogni nodo della rete, fatta eccezione per la sorgente s e per il pozzo t, la quantità totale di flusso in ingresso deve uguagliare esattamente la quantità totale di flusso in uscita. Ciò impone l'assenza di accumulo, dispersione o generazione spontanea nei nodi intermedi (Principio di Kirchhoff per le correnti):

$$\sum_{v \in V:(u,v)\in A} f(u,v) - \sum_{v \in V:(v,u)\in A} f(v,u) = 0 \quad \forall u \in V \setminus \{s,t\} \tag{2.3}$$

Il problema del **Flusso Massimo** (*Maximum Flow Problem*) consiste nel determinare un instradamento ammissibile del flusso f che massimizzi il valore del flusso netto totale $v(f)$ uscente dalla sorgente s e diretto verso il pozzo t. Poiché le relazioni che governano il sistema sono interamente lineari, il problema può essere formulato come un modello di **Programmazione Lineare (PL)**:

$$\max \quad v(f) = \sum_{v \in V:(s,v)\in A} f(s,v) \tag{2.4}$$

$$\text{s.t.} \quad f(u,v) \leq c(u,v) \quad \forall (u,v) \in A \tag{2.5}$$

$$\sum_{v \in V:(u,v)\in A} f(u,v) - \sum_{v \in V:(v,u)\in A} f(v,u) = 0 \quad \forall u \in V \setminus \{s,t\} \tag{2.6}$$

$$f(u,v) \geq 0 \quad \forall (u,v) \in A \tag{2.7}$$

La risoluzione di questo modello fornisce non solo il valore numerico del massimo throughput della rete, ma anche l'assegnazione ottima delle risorse (il flusso) su ciascun arco della topologia.

2.1.3 Tagli e il Teorema Max-Flow Min-Cut

Un concetto duale e inscindibile da quello del flusso massimo è quello del **taglio** di una rete. Comprendere la nozione di taglio è cruciale per analizzare la vulnerabilità e i "colli di bottiglia" strutturali di un grafo.

Si definisce **taglio** $s-t$ in una rete di flusso $G=(V,A)$ una partizione dell'insieme dei vertici V in due sottoinsiemi disgiunti S e T tali che:

$$s \in S \quad \text{e} \quad t \in T \quad \text{con} \quad S \cup T = V, \quad S \cap T = \emptyset \tag{2.8}$$

La **capacità di un taglio** (S,T), denotata con $C(S,T)$, è definita come la somma delle capacità di tutti gli archi che hanno origine in un nodo dell'insieme S e destinazione in un nodo dell'insieme T:

$$C(S,T) = \sum_{u \in S, v \in T : (u,v) \in A} c(u,v) \tag{2.9}$$

Si noti che gli archi orientati nel verso opposto, ovvero da T ad S, non contribuiscono alla capacità del taglio.

Un **taglio minimo** è un taglio la cui capacità $C(S,T)$ è la minima possibile tra i tagli ammissibili che separano s da t. L'importanza capitale di questa entità matematica risiede in uno dei teoremi più celebri dell'ottimizzazione combinatoria, dimostrato indipendentemente da P. Elias, A. Feinstein, e C. Shannon, e da L.R. Ford e D.R. Fulkerson nel 1956.

Teorema 1 (Max-Flow Min-Cut) *In una generica rete di flusso G, il valore del flusso massimo trasmissibile dalla sorgente s al pozzo t è esattamente uguale alla capacità del taglio minimo $s-t$ della rete.*

$$\max v(f) = \min_{S,T} C(S,T) \tag{2.10}$$

L'intuizione fisica alla base del teorema è di una chiarezza disarmante: la massima quantità di materia che può defluire da un punto ad un altro di un sistema complesso è interamente limitata dalla sezione più stretta dell'intero sistema. Gli archi che costituiscono il taglio minimo rappresentano a tutti gli effetti il **collo di bottiglia strutturale** (*bottleneck*) della rete. Se questi specifici archi dovessero essere saturati o rimossi, la comunicazione tra s e t risulterebbe fisicamente impossibile.

2.1.4 L'Algoritmo di Edmonds-Karp e l'Analisi della Complessità

Seppur il modello PL (2.4-2.7) sia risolvibile con algoritmi general purpose come il metodo del Simplesso, la struttura peculiare del problema permette l'impiego di algoritmi combinatori enormemente più efficienti. Tra questi spicca il celebre **Algoritmo di Edmonds-Karp** [1], proposto nel 1972 come una specializzazione deterministica del metodo di Ford-Fulkerson.

Il metodo generico di Ford-Fulkerson si basa sulla ricerca iterativa di **cammini aumentanti** all'interno di una **rete residua** $G_f = (V, A_f)$. Data la rete G e un flusso corrente f, la rete residua rappresenta la capacità residua ancora sfruttabile. Per ogni arco $(u, v) \in A$:

- La capacità residua in avanti (quanto flusso addizionale può essere spinto) è $r(u, v) = c(u, v) - f(u, v)$.
- La capacità residua all'indietro (quanto flusso può essere "ritirato" o ri-direzionato) è pari al flusso attuale $r(v, u) = f(u, v)$.

L'algoritmo procede trovando un cammino da s a t in G_f (il cammino aumentante) in cui ogni arco possiede capacità residua strettamente positiva. Si determina il minimo tra le capacità residue lungo tale cammino (chiamato *collo di bottiglia del cammino*) e si incrementa il flusso globale di tale quantità, aggiornando le capacità residue di conseguenza. Il processo termina quando non esiste alcun cammino ammissibile in G_f tra s e t.

Un limite del metodo originale di Ford-Fulkerson risiede nell'arbitrarietà con cui il cammino aumentante viene scelto ad ogni iterazione. Se la selezione è inefficiente, e specialmente in presenza di capacità con valori irrazionali, l'algoritmo può effettuare una serie infinita di incrementi infinitesimi senza mai convergere al valore ottimo. Nel caso peggiore con capacità intere, la complessità è proporzionale al valore del flusso massimo stesso: $O(E \cdot |f^*|)$, non garantendo dunque un tempo polinomiale rispetto alle dimensioni dell'input.

Edmonds e Karp hanno risolto brillantemente questo problema introducendo un'euristica di selezione vincolante: ad ogni iterazione, **il cammino aumentante scelto deve essere il cammino minimo**, inteso in termini di numero minore di archi (pesi unitari), tra la sorgente e il pozzo. Questa operazione di ricerca viene effettuata implementando il classico algoritmo di esplorazione non informata **Breadth-First Search (BFS)**.

L'uso della BFS per la selezione del cammino offre proprietà matematiche di convergenza formidabili. Si dimostra infatti che:
1. La distanza (in numero di archi) tra s ed ogni altro nodo $u \in V$ nella rete residua $d_f(s, u)$ non decresce mai monotònicamente durante l'esecuzione dell'algoritmo.
2. Ogni volta che un arco viene saturato ("arco critico") in una BFS e poi riappare in seguito in G_f con direzione opposta, la distanza del suo nodo sorgente rispetto ad s deve essere aumentata di almeno due unità.
3. Poiché la distanza massima in un grafo di V nodi è $V - 1$, un arco può diventare critico al massimo $V/2$ volte.
4. Poiché ad ogni iterazione almeno un arco diventa critico (scomparendo da G_f), il limite superiore teorico sul numero totale di iterazioni prima che l'algoritmo termini è rigorosamente limitato a $O(V \cdot E)$.

Considerando che una singola passata di BFS esplora ogni arco al massimo una volta, impiegando un tempo $O(E)$, la **complessità computazionale asintotica**

nel caso peggiore dell'algoritmo di Edmonds-Karp si fissa saldamente al valore strettamente polinomiale di:

$$O(V \cdot E^2) \tag{2.11}$$

Questo limite rende l'algoritmo incredibilmente rapido ed affidabile, svincolandolo completamente dal valore numerico delle capacità e rendendolo idoneo a valutazioni in real-time in scenari applicativi complessi.

2.1.5 Teoria delle Clique e l'Algoritmo di Bron-Kerbosch

Analogamente al problema del flusso su grafi orientati, un'altra colonna portante della Ricerca Operativa riguarda l'analisi della coesione strutturale e della mutua copertura all'interno di grafi **non orientati**. Questo tipo di analisi è dominato dalla teoria delle **Clique** (cricche).

Sia $G' = (V, E')$ un grafo semplice non orientato, dove E' rappresenta l'insieme degli spigoli privi di direzione.

- Si definisce **Clique** un sottoinsieme di nodi $C \subseteq V$ tale per cui il sottografo indotto da C risulta completo. Geometricamente e topologicamente, questo significa che per ogni singola coppia di nodi distinti appartenenti a C, esiste necessariamente uno spigolo che li connette direttamente:

$$\forall u, v \in C \quad \text{con } u \neq v \implies \{u, v\} \in E' \tag{2.12}$$

- Una clique C si dice **Massimale** se è impossibile aggiungervi un ulteriore nodo appartenente a $V \setminus C$ senza violarne la proprietà di completezza (ovvero, nessun altro nodo esterno è connesso a *tutti* i nodi attualmente presenti in C).

- Una clique C si dice **Massima** se è di massima cardinalità tra tutte le clique massimali presenti nel grafo. Il numero di nodi che la compongono definisce il *numero di clique* del grafo, denotato con il simbolo $\omega(G')$.

Trovare la clique massima $\omega(G')$ di una rete è equivalente a individuare il sottogruppo più esteso di agenti (o entità) che condividono tra loro, in modo esaustivo e bilaterale, una certa proprietà di vicinanza o cooperazione totale. Il calcolo di questo parametro costituisce tuttavia uno dei classici e più famosi problemi **NP-difficili** (NP-hard) della teoria della complessità computazionale formulati da Richard Karp. La versione decisionale associata del problema appartiene alla classe dei problemi **NP-completi**.

Non essendo ad oggi noti algoritmi capaci di risolverlo in tempo polinomiale deterministico, l'approccio esatto più efficiente ed universalmente riconosciuto in letteratura per grafi sparsi è l'algoritmo di **Bron-Kerbosch** [2].

Questo algoritmo esplora l'intero spazio delle soluzioni avvalendosi della tecnica del *backtracking* ricorsivo, manipolando e confrontando simultaneamente tre insiemi disgiunti di nodi:

1. R: I nodi che compongono la clique in via di formazione nel ramo corrente.
2. P: I nodi "prospettici", ovvero i candidati validi che sono adiacenti a *tutti* i nodi attualmente in R e che potrebbero quindi estendere la clique.
3. X: I nodi esclusi. Si tratta di nodi che estenderebbero validamente R, ma che sono già stati esplorati esaustivamente in rami ricorsivi precedenti. Il loro scopo è puramente potativo: prevengono che l'algoritmo generi cloni (permutazioni) della medesima clique massimale, risparmiando un'enorme mole di calcoli.

Lo pseudocodice classico è il seguente:

Procedura BronKerbosch(R, P, X)
1. **if** $P = \emptyset$ **and** $X = \emptyset$ **then**
2. La clique corrente in R è **massimale**. Memorizzala.
3. **end if**
4. **for each** vertice $v \in P$ **do**
5. Inizia un nuovo ramo includendo v:
6. BronKerbosch($R \cup \{v\}$, $P \cap \Gamma(v)$, $X \cap \Gamma(v)$)
7. Sposta v nei nodi esplorati per i prossimi cicli di questo livello:
8. $P \leftarrow P \setminus \{v\}$
9. $X \leftarrow X \cup \{v\}$
10. **end for**

dove $\Gamma(v)$ indica l'intorno del nodo v. La versione avanzata dell'algoritmo, fondamentale per le performance, introduce l'euristica del **Pivoting**. Scegliendo un nodo pivot strategico $u \in P \cup X$, il ciclo for alla linea 4 può essere limitato esclusivamente ai nodi non adiacenti ad u (ovvero ai vertici in $P \setminus \Gamma(u)$), minimizzando enormemente la ramificazione dell'albero di ricerca e garantendo risposte rapide persino in contesti di elaborazione iterativa in real-time.

2.2 Giochi su Reti e Network Interdiction

Sebbene l'algoritmo di Edmonds-Karp descriva il comportamento di flussi passivi che cercano un instradamento ottimale, molti contesti reali sono dominati dalla presenza di entità attive ed intelligenti che operano in diretta competizione. La Ricerca Operativa modella queste dinamiche avversarie tramite la disciplina dei *Network Games* e dei complessi modelli di *Network Interdiction*.

2.2.1 Introduzione ai Network Games

I Giochi su Reti (*Network Games*) rappresentano un connubio tra la teoria dei giochi classica di Nash e von Neumann, e la topologia strutturale fornita dalla teoria dei grafi. In tali modelli, la strategia ottima e il *payoff* (la ricompensa) di ciascun agente o decisore non dipendono dall'interazione globale con l'intero sistema, bensì

sono fortemente vincolati dalla specifica posizione topologica e dalla connettività locale (l'intorno) in cui l'agente è situato.

Queste simulazioni prevedono che le risorse siano limitate e che l'utilità di un agente diminuisca all'aumentare dell'ostruzionismo operato dagli avversari. Una branca affascinante dei Network Games riguarda l'analisi in cui il campo di battaglia non è statico, ma l'obiettivo principale di una fazione è l'alterazione fisica del grafo stesso: modificare, danneggiare o proteggere i nodi e gli archi per deviare i flussi o frammentare la comunicazione del nemico.

2.2.2 I Modelli Bi-Livello di Network Interdiction

Questa manipolazione antagonista della rete prende il nome di problema di interdizione su reti (*Network Interdiction*). Esso costituisce una classe di estrema importanza di problemi di ottimizzazione matematica bi-livello (*Bilevel Optimization*), strutturati tipicamente come un Gioco di **Stackelberg** non cooperativo a informazione perfetta.

Il framework canonico prevede una sequenza temporale asimmetrica che coinvolge due attori razionali:

1. Il **Leader** (Interdittore o Attaccante): gioca per primo e dispone di un budget limitato di risorse o manodopera che impiega per "attaccare" componenti critici della rete. Il suo attacco mira a distruggere (rimuovere archi) o degradare (ridurre la capacità) l'infrastruttura nemica, anticipando le contromosse.

2. Il **Follower** (Operatore o Difensore): osserva l'attacco compiuto dal Leader (la nuova topologia alterata o residua della rete) e gioca per secondo. Egli risolve un tradizionale problema di ottimizzazione (ad es. massimizzare il flusso o trovare il cammino di costo minimo) per trarre il massimo vantaggio da ciò che resta del suo sistema.

L'obiettivo strategico primario del Leader è selezionare un piano di interdizione che minimizzi l'utilità massima (il risultato dell'ottimizzazione) che il Follower potrà estrarre dal grafo superstite. Nel celebre problema originario del Maximum Flow Network Interdiction (MFNI) introdotto da Kevin Wood [3], il modello è formulato come un problema di ottimizzazione "min-max":

$$\min_{x \in X} \quad \max_{f} \quad v(f) = \sum_{v \in V:(s,v) \in A} f(s,v) \tag{2.13}$$

$$\text{s.t.} \quad f(u,v) \leq c(u,v) \cdot (1 - x_{u,v}) \quad \forall (u,v) \in A \tag{2.14}$$

$$\sum_{v \in V:(u,v) \in A} f(u,v) - \sum_{v \in V:(v,u) \in A} f(v,u) = 0 \quad \forall u \in V \setminus \{s,t\} \tag{2.15}$$

$$f(u,v) \geq 0 \quad \forall (u,v) \in A \tag{2.16}$$

Nelle equazioni (2.13)-(2.16):

- $x_{u,v} \in \{0,1\}$ è la variabile decisionale binaria del leader. Se $x_{u,v} = 1$, l'arco (u,v) è considerato attaccato/distrutto con successo. La sua capacità effettiva nella disequazione (2.14) viene forzata a zero, annullando il flusso passante.
- $X \subseteq \{0,1\}^{|A|}$ definisce lo spazio delle configurazioni di attacco ammissibili, che è sempre vincolato da una risorsa cumulativa:

$$\sum_{(u,v)\in A} r_{u,v} x_{u,v} \leq B \tag{2.17}$$

  dove $r_{u,v}$ è lo sforzo speso per distruggere l'arco (u,v) e B è il budget totale del leader.

Oltre a questi modelli estremi di interdizione binaria, la ricerca accademica moderna si sta spostando verso l'**Interdizione Parziale (o riduttiva)**. In scenari in cui la distruzione totale di un nodo non è fisicamente plausibile (come nell'opposizione probabilistica in un evento sportivo o nel *jamming* di un segnale wireless), l'interdizione agisce non per rimuovere l'arco, ma per imporre una penalità $0 < \alpha_{u,v} < 1$ che scalfisce la capacità originaria:

$$c_{residua}(u,v) = c(u,v) \cdot (1 - \alpha_{u,v}) \tag{2.18}$$

Tale formulazione incrementa vertiginosamente la complessità analitica, ma modella con un grado di precisione incomparabile i sistemi stocastici del mondo reale.

2.3 I Serious Games in Ambito Didattico

L'impianto teorico descritto finora (grafi, cammini, tagli minimi, flussi ed interdizione di Stackelberg) costituisce un bagaglio scientifico di elevata astrazione. Spesso gli studenti universitari affrontano l'apprendimento di tali tematiche in forma esclusivamente algebrica o procedurale. La trasposizione di tali astrazioni in ambienti dinamici interattivi non è solo un esercizio tecnologico, ma un'innovazione pedagogica di forte spessore, che rientra nel dominio accademico dei *Serious Games*.

2.3.1 Definizione e Scopi Didattici

Il termine *Serious Game* è stato coniato formalmente nel 1970 da Clark C. Abt [4], pioniere dei giochi di simulazione formativa. Egli li descrive come:

> «[...] giochi accuratamente pianificati che possiedono una finalità esplicita di formazione, istruzione ed educazione, e che si distaccano strutturalmente dai software concepiti esclusivamente per il puro e semplice intrattenimento.»

Negli ultimi due decenni, la convergenza tra la potenza di calcolo dei motori grafici (Game Engines) e gli studi cognitivi ha posizionato i Serious Games come strumenti

cruciali per le discipline STEM. L'efficacia di tali strumenti poggia su tre pilastri cardine della pedagogia costruttivista:

- **Apprendimento Esperienziale e Problem-Based Learning:** L'utente non è più un recettore passivo di formule dimostrate su una lavagna. Inserito in un ambiente di gioco limitato da regole, il fruitore esplora autonomamente le variabili, formula ipotesi e le testa in modo empirico e non rischioso, arrivando a comprendere la soluzione come diretta conseguenza della propria strategia logica.
- **Ciclo di Feedback Immediato (*Feedback Loop*):** Nei sistemi matematici formali, risalire agli effetti di una variazione marginale di un parametro può richiedere ore di rielaborazione analitica. In una simulazione, modificare l'intensità di un flusso o eliminare un arco comporta un aggiornamento computazionale visibile a schermo in frazioni di secondo (grazie all'efficienza degli algoritmi studiati in precedenza), cementificando la comprensione causa-effetto.
- **Teoria del Flusso e *Engagement*:** Il gioco maschera l'eventuale frustrazione nell'apprendimento tramite l'introduzione di obiettivi stimolanti (battere l'avversario), trasportando lo studente in uno stato mentale di estremo coinvolgimento (il *Flow* descritto dallo psicologo Csikszentmihalyi), massimizzando al contempo l'inconscia ritenzione mnemonica della teoria.

2.3.2 La Metodologia del Mascheramento Algoritmico

Il paradigma d'avanguardia per l'impiego didattico della Ricerca Operativa prende il nome di **Mascheramento Algoritmico** (*Algorithmic Shrouding*). Questa tecnica consiste nel "nascondere" i tecnicismi analitici del problema sotto forma di metafore visive tridimensionali altamente comprensibili, permettendo all'intuizione spaziale umana di processare concetti alla base di problemi NP-difficili o complessi senza subire il sovraccarico cognitivo del puro formalismo.

Questo approccio si rivela rivoluzionario:

- La **capacità di un canale** $c(u, v)$ non è un semplice intero stampato a console, ma viene percepita visivamente attraverso lo spessore dell'arco che si gonfia in tempo reale.
- I **limiti fisici e i vincoli vettoriali** assumono la forma di geometrie proiettate direttamente nello spazio tridimensionale navigabile.
- L'**interdizione su rete** cessa di essere una disuguaglianza in un modello min-max bi-livello, trasformandosi visivamente nella lotta tra la pressione esercitata dal difensore e la resistenza offerta dai flussi offensivi.

L'applicazione ludica diventa quindi un vero e proprio laboratorio computazionale virtuale. Lo studente può sfidare se stesso nell'identificare ad occhio nudo il taglio

minimo di una rete, comprendere l'importanza critica dei colli di bottiglia e interiorizzare la logica di ottimizzazione prima ancora che gli venga mai presentata la sua rigida dimostrazione analitica sulla carta, chiudendo il cerchio tra teoria formale e pura intuizione geometrica.

Capitolo 3

OptiSoccer: Architettura e Modello di Gioco

In questo capitolo viene presentata la modellazione logica, l'architettura delle regole e la strutturazione software del videogioco **OptiSoccer**, un *serious game* sviluppato nell'ambiente di simulazione tridimensionale Unity. L'obiettivo centrale è illustrare come i concetti astratti della Teoria dei Grafi, della Ricerca Operativa e della programmazione lineare descritti nel Capitolo 2 siano stati concretamente ingegnerizzati in una simulazione interattiva, capace di tradurre l'ottimizzazione formale in strategie di gameplay ludico ed educativo.

3.1 Architettura Software in Unity

L'infrastruttura software di OptiSoccer è stata progettata seguendo il paradigma architetturale a componenti (*Component-Based Architecture*), nativo del motore Unity. Questo paradigma separa rigidamente i dati strutturali, la logica di controllo e la rappresentazione visiva, garantendo un'elevata scalabilità del sistema in fase di simulazione algoritmica.

3.1.1 Gestione dei Dati Tramite ScriptableObjects

Nel mondo reale, le proprietà e le statistiche dei calciatori rappresentano un dataset denso ed eterogeneo. Per evitare ridondanze in memoria e migliorare le performance di istanziazione, OptiSoccer implementa un approccio basato sugli **ScriptableObjects** (SO). Un SO è un contenitore di dati nativo di Unity che permette di salvare enormi quantità di informazioni slegate dalle istanze specifiche presenti nella scena (i*GameObject*).

Ogni entità logica del gioco (come il profilo di un atleta, le formazioni standard e i parametri dei ruoli) è definita come uno ScriptableObject e salvata fisicamente come un file *.asset* autonomo nel database di progetto. L'uso degli ScriptableObject in un serious game di Ricerca Operativa è cruciale per le performance: quando si instanziano centinaia di archi e decine di nodi per costruire e distruggere grafi multipli durante la ricerca del flusso massimo o delle cricche, l'assenza di dati duplicati riduce drasticamente l'impatto sul *Garbage Collector* e preserva l'efficienza

temporale richiesta ($O(V \cdot E^2)$) dalla BFS di Edmonds-Karp, permettendo al motore di mantenere i 60 frame per secondo.

3.1.2 Struttura dei Nodi e degli Archi

La rete di flusso è tradotta a schermo tramite due classi fondamentali:

- `PlayerNode.cs`: Controlla la logica del singolo nodo. Legge i dati invarianti dal proprio 'PlayerStatsSO' (lo ScriptableObject associato) e definisce il comportamento interattivo (ad esempio, le funzionalità di trascinamento tramite puntatore del mouse). Conserva la sua posizione tridimensionale aggiornata, essenziale per il calcolo euristico e probabilistico dell'interdizione.
- `GraphEdge.cs`: Incapsula la logica degli archi orientati. Un arco è rappresentato visivamente come una geometria spline 3D che unisce la sorgente `nodeA` alla destinazione `nodeB`. L'arco conserva lo stato dinamico della propria **capacità** $c(u, v)$ e del flusso transitante, esibendo un *shader* visivo parametrico: lo spessore cilindrico del modello 3D e l'emissione luminosa scalano linearmente o passano da un colore neutro al colore di squadra in misura direttamente proporzionale alla percentuale di utilizzo $\left(\frac{f(u,v)}{c(u,v)}\right)$.

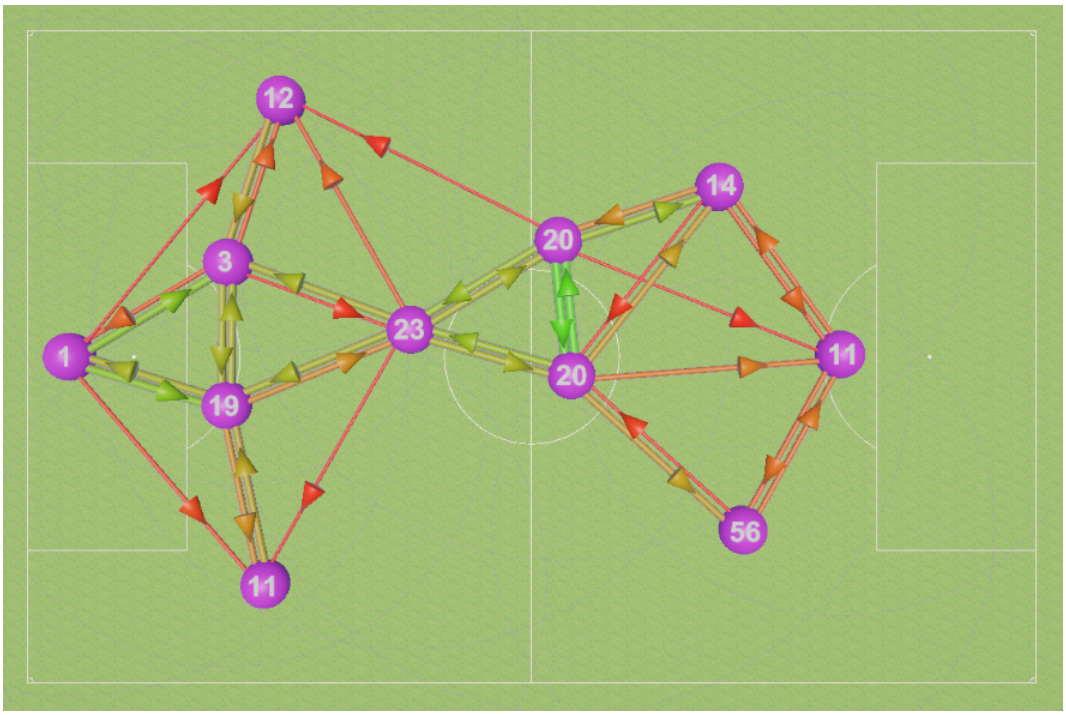

Figura 3.1: Visualizzazione bidimensionale di OptiSoccer. Le linee colorate rappresentano gli archi orientati capacitivi generati tra i nodi (i calciatori) in funzione del loro raggio d'azione.

3.2 Il Modello del Grafo Calcistico

OptiSoccer è concepito come un simulatore tattico il cui nucleo non risiede nella fisica della palla, bensì nell'esplorazione e nella manipolazione di una complessa rete di instradamento.

La corrispondenza formale tra la terminologia calcistica e la Teoria dei Grafi è la seguente:

- **Nodi (Calciatori):** Ciascun calciatore $v_i \in V$ rappresenta un nodo del grafo. Ogni nodo è dotato di un proprio stato interno costituito da statistiche atletiche e spaziali specifiche.
- **Archi Orientati (Affinità e Interazione):** Un passaggio potenziale tra il calciatore u e il calciatore v definisce un arco orientato $(u, v) \in A$. L'arco è attivo se e solo se la distanza euclidea spaziale tra i due nodi non supera il raggio di passaggio efficace della sorgente ($d(u, v) \leq R_{eff}(u)$).
- **Capacità dell'Arco:** La capacità base $c(u, v)$ è determinata dalla combinazione delle statistiche tecniche della sorgente u (visione di gioco, potenza) e dalla facilità di ricezione del nodo destinazione v. Maggiore è la capacità, maggiore sarà la quota di flusso offensivo che la squadra potrà veicolare lungo quella specifica direttrice, rendendo la rete più tollerante all'interdizione.

All'interno di questo grafo orientato, vengono identificati due nodi essenziali per la costituzione di una valida rete di flusso:

1. **Sorgente (s):** Il portiere della squadra (*Goalkeeper*) funge da sorgente fissa. La manovra offensiva si origina necessariamente da questo nodo, che immette flusso puro nella rete. A livello topologico, il portiere è vincolato spazialmente e non può superare la propria trequarti difensiva.
2. **Pozzo (t):** Il centravanti avanzato (*Center Forward*) funge da pozzo del grafo, assorbendo il flusso veicolato dall'intera squadra per finalizzare l'azione e "segnare". Per riflettere il comportamento di una punta di sfondamento, lo slot del pozzo è tipicamente posizionato e vincolato nell'ultimo quinto di campo.

3.3 Creazione della Squadra e Problema dello Zaino

La fase antecedente la partita giocata prende il nome di **Draft** (Mercato o Selezione). Durante questa fase, l'obiettivo cognitivo e matematico è la selezione strategica degli 11 nodi che costituiranno la rete della propria formazione titolare, attingendo da un database di oltre 100 atleti storici.

Questa fase è una fedele riproduzione del **Problema dello Zaino Multi-Vincolo** (*Multi-Constraint Knapsack Problem*). La variante implementata in OptiSoccer è specificatamente una versione di **0-1 Knapsack** (zaino zero-uno), poiché le frazioni

di giocatori non sono permesse: un atleta reale deve essere inserito interamente nella rete ($x_i = 1$) o scartato ($x_i = 0$).

Il modello matematico del Draft è formulato come segue. Sia D il database complessivo dei calciatori disponibili. Per ciascun calciatore $i \in D$, sono noti il costo finanziario in crediti C_i, il reparto di appartenenza e le statistiche prestazionali. Definiamo $x_i \in \{0, 1\}$ la variabile decisionale binaria associata alla scelta dell'atleta.

Il problema dell'ottimizzazione della propria squadra in pre-partita si pone come:

$$\max \quad \sum_{i \in D} U_i x_i \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in D} C_i x_i \leq B \tag{3.2}$$

$$\sum_{i \in D} x_i = 11 \tag{3.3}$$

$$\sum_{i \in D_{POR}} x_i = 1 \tag{3.4}$$

$$\sum_{i \in D_{SINK}} x_i = 1 \tag{3.5}$$

$$x_i \in \{0, 1\} \quad \forall i \in D \tag{3.6}$$

dove:

- L'equazione (3.1) mira a massimizzare l'utilità complessiva della rete. U_i rappresenta la somma aggregata delle prestazioni base dell'atleta i.
- La disequazione (3.2) modella il rigoroso vincolo di budget finanziario o *Salary Cap*: la somma totale dei costi non deve superare il budget limite imposto (B).
- Le restanti equazioni impongono vincoli tattici e logici invalicabili: la squadra deve formarsi in modo canonico con un solo portiere (che genera sorgente) ed un unico nodo designato come *pozzo*.

Il costo monetario C_i di ciascun calciatore è intrinsecamente ancorato ai suoi parametri operativi tramite l'equazione deterministica:

$$C_i = (PassRange_i + MaxCapacity_i + AttackPower_i + DefensePower_i) \times 5 \tag{3.7}$$

Questa relazione lineare tra statistiche fisiche e vincolo monetario è deliberatamente progettata per forzare lo studente (il giocatore) ad affrontare complessi *trade-off* tipici dell'ottimizzazione combinatoria non frazionaria. La presenza di un costo proporzionale rende matematicamente inattuabile l'acquisto di una squadra di soli fuoriclasse (poiché si violerebbe (3.2)), spingendo il giocatore ad affiancare colli di bottiglia (nodi a basso costo e bassa capacità) in punti in cui si ritiene che l'interdizione avversaria non focalizzerà i propri sforzi.

La Tabella 3.1 evidenzia alcuni esempi pratici di costo e relative statistiche.

Tabella 3.1: Estratto del database atleti: Statistiche fisiche, vincoli e costi associati

Nome Calciatore	Ruolo	PassRange	MaxCapacity	AttackPower	DefensePower	Costo (€)
G. Buffon	POR	12.0	8.0	1.0	10.0	155
G. Chiellini	DC	16.0	6.0	3.0	8.0	165
A. Pirlo	CM	24.0	6.0	5.0	5.0	200
F. Totti	TRQ	20.0	6.0	6.0	4.0	180
A. Del Piero	ATT	16.0	7.0	8.0	3.0	170

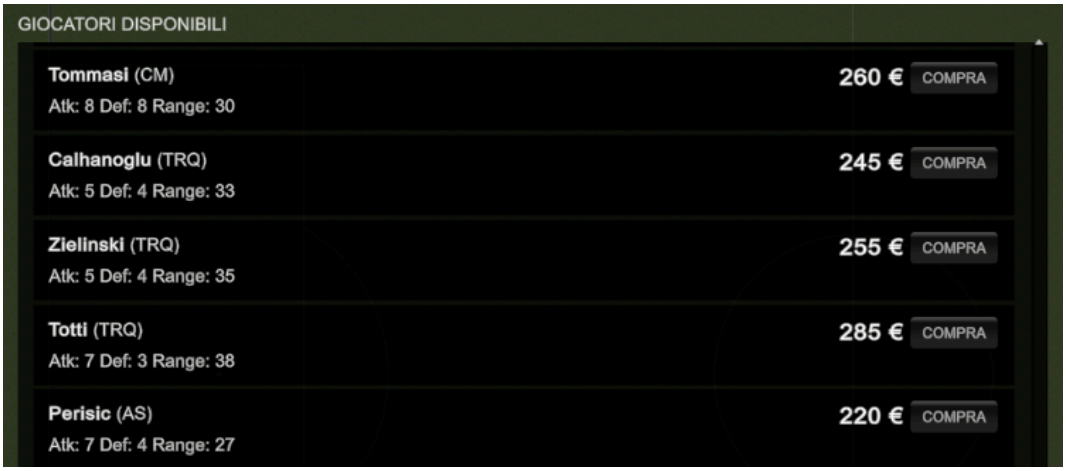

Figura 3.2: Interfaccia utente per la risoluzione pratica dello Zaino Multi-Vincolo durante la fase di Draft di OptiSoccer.

3.4 Dinamiche Spaziali e Decadimento Cubico (Affinity Zone)

In OptiSoccer la topologia del grafo dipende dall'embedding dei nodi nel piano. Il giocatore dispone i nodi sul campo virtuale bidimensionale, ma non può posizionarli indiscriminatamente. Il concetto calcistico della "disciplina tattica" viene formalizzato tramite l'**Affinity Multiplier 2D**.

Ogni classe di ruolo calcistica possiede un'area di competenza intrinseca, geometricamente definita come una finestra rettangolare normalizzata $W = [x_{min}, x_{max}] \times [z_{min}, z_{max}]$. All'interno di quest'area, il giocatore è nel proprio "habitat" tattico e il suo rendimento (il raggio di espansione dell'arco $PassRange$) si esprime al 100%. Uscendo da tale zona, il raggio di trasmissione viene troncato progressivamente da una penalizzazione.

Sia (x_{norm}, z_{norm}) la coordinata normalizzata proiettata del nodo rispetto al campo di gioco e alla direzione d'attacco. Si calcolano le distanze euclidee lineari di deviazione rispetto ai bordi della zona di comfort W:

$$d_z = \max(0, z_{min} - z_{norm}, z_{norm} - z_{max}) \cdot L_{field} \tag{3.8}$$

$$d_x = \max(0, x_{min} - x_{norm}, x_{norm} - x_{max}) \cdot W_{field} \tag{3.9}$$

Lo sforamento complessivo netto in metri fisici è pari a $d_{total} = \sqrt{d_x^2 + d_z^2}$.

La penalizzazione imposta ai giocatori che si trovano "fuori posizione" non è affatto lineare. Per simulare la drastica e improvvisa disconnessione causata dallo smarrimento spaziale dell'atleta, viene introdotta una **funzione cubica di attenuazione (cubic ease-out falloff)**.

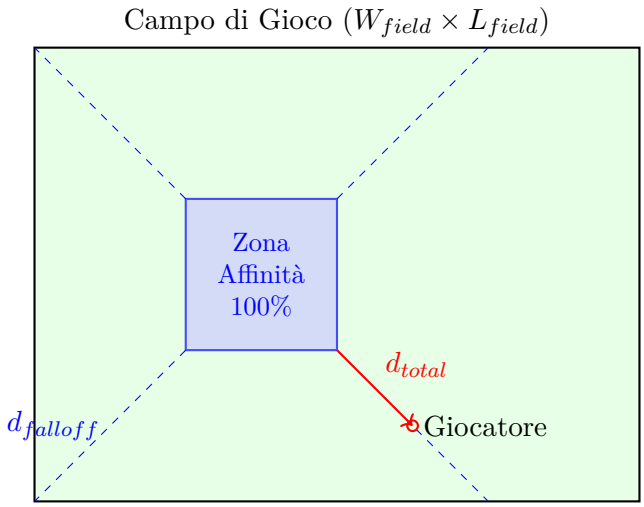

Figura 3.3: Rappresentazione geometrica dell'Affinity Zone. Il rettangolo blu rappresenta l'area in cui $d_{total} = 0$ e $A_{mult} = 1.0$. Allontanandosi da essa, la distanza d_{total} fa decrescere il moltiplicatore secondo una curva cubica.

Posto $d_{falloff}$ il raggio limite di tolleranza per quel ruolo, la variabile adimensionale di uscita si calcola come $t = \min\left(1, \frac{d_{total}}{d_{falloff}}\right)$. L'applicazione del filtro non lineare restituisce la curva di mitigazione t_{curve}:

$$t_{curve} = 1 - (1 - t)^3 \tag{3.10}$$

Tale operatore non lineare restituisce il **Moltiplicatore di Affinità** finale ($A_{mult} \in [A_{min}, 1.0]$) per interpolazione:

$$A_{mult} = 1.0 - t_{curve} \cdot (1.0 - A_{min}) \tag{3.11}$$

$$R_{eff} = PassRange \cdot A_{mult} \tag{3.12}$$

Questa equazione ha riflessi enormi sulla robustezza del grafo finale che l'utente sta tentando di tracciare. Costringere ad esempio un terzino difensivo (che dispone tipicamente di un'area di comfort stretta e limitata al proprio quadrante) a schierarsi a ridosso dell'attacco, produrrà un valore di t unitario o prossimo ad esso. Elevando la deviazione al cubo, il moltiplicatore collasserà quasi istantaneamente al valore critico minimo ammissibile A_{min} (solitamente fissato al 10%). Il raggio di passaggio effettivo crollerà a frazioni di metro, impedendo la generazione del corrispondente arco orientato e isolando completamente il nodo dalla rete. Tale penalità modella

formalmente l'estrema importanza del corretto impiego tattico.

Capitolo 4

Algoritmi di Valutazione e Risoluzione

In questo capitolo viene esaminato il motore di calcolo algoritmico sottostante a **OptiSoccer**. Mentre i capitoli precedenti hanno illustrato la struttura logica e geometrica pre-partita, qui viene affrontata l'esecuzione della fase di gioco attiva: il momento dello scontro fisico ed intercettazione dei passaggi (*Combat*), seguito dall'immediata misurazione topologica della rete risultante. Particolare rilievo verrà dato all'analisi matematica del comportamento asintotico delle formule probabilistiche utilizzate, alle scelte architetturali in C# per garantire l'affidabilità numerica e all'ottimizzazione euristica dei problemi NP-difficili per l'esecuzione fluida in *Real-Time*.

4.1 Il Sistema di Combat Simultaneo

La fase di **Combat** rappresenta il fulcro empirico del modello di *Network Interdiction* analizzato teoricamente nel Capitolo 2. In molti modelli teorici di ricerca operativa (es. nei giochi di Stackelberg), il problema bi-livello globale viene risolto matematicamente da un unico solutore centralizzato. Per replicare invece la dinamica agonistica e distribuita di una partita di calcio, *OptiSoccer* adotta un approccio decentralizzato ad *agenti autonomi*. Ciascun giocatore schierato sul campo valuta l'ambiente circostante e agisce in modo indipendente per degradare o rimuovere archi specifici del grafo avversario.

L'esecuzione del combattimento è affidata al modulo di simulazione interna 'CombatResolver' ed è processata in una sequenza rigorosa di fasi spaziali e probabilistiche.

4.1.1 Geometria dell'Area di Pressing e Selezione dei Bersagli

Ogni difensore presente sul terreno di gioco proietta attorno a sé una **zona di pressing** circolare, il cui raggio corrisponde direttamente all'efficacia del suo posizionamento tattico:

$$Range_{press} = R_{eff} = PassRange \cdot A_{mult} \tag{4.1}$$

Per ciascun difensore D, il motore geometrico scandaglia l'intero set degli archi di passaggio attivi generati dalla squadra in attacco. Siano p_u e p_v le coordinate

spaziali sul piano xz del campo dei nodi sorgente e destinazione di un arco orientato avversario (u, v). Tale arco viene classificato come **attaccabile** (ovvero entra nel radar di intercettazione di D) se almeno una delle sue estremità (il tiratore o il ricevitore) cade all'interno del disco di pressing del difensore posizionato in p_D:

$$\min\left(d(p_D, p_u), d(p_D, p_v)\right) \leq Range_{press} \tag{4.2}$$

dove $d(p_A, p_B) = \sqrt{(x_A - x_B)^2 + (z_A - z_B)^2}$ rappresenta la metrica euclidea standard.

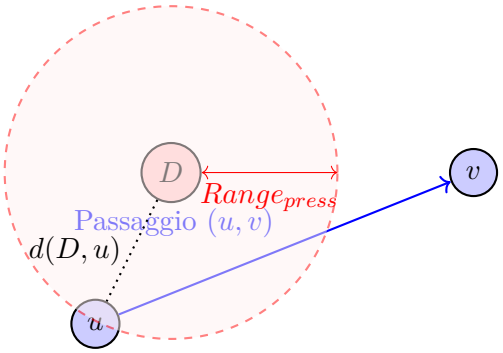

Figura 4.1: Intersezione tra il raggio di pressing del difensore (rosso) e un arco di passaggio avversario (blu). Poiché $d(D, u) \leq Range_{press}$, l'arco (u, v) è classificato come attaccabile.

Nella maggior parte delle fasi di gioco congestionate, un difensore rileverà un elevato numero di archi intercettabili, creando un problema di allocazione ottimale delle proprie risorse fisiche. Il software simula la limitazione dell'attenzione umana e della mobilità istantanea applicando una **politica di priorità spaziale**:

1. Il difensore calcola la distanza euclidea minima che lo separa da ogni arco valido.

2. L'intera lista di bersagli viene ordinata in modo crescente (i pericoli tatticamente più prossimi hanno priorità assoluta).

3. Il difensore attacca i primi k archi della lista, dove $k = MaxTargets$ è una statistica del database (tipicamente 1 per i centravanti che pressano, 2 per i mastini di centrocampo o difensori centrali). Tutti gli altri archi restano liberi dall'influenza di D per quel turno.

4.1.2 Formula di Interdizione

Una volta che l'arco bersaglio è stato acquisito, il difensore tenta il tackle o l'intercettazione. L'esito di questa operazione non è deterministico: il sistema risolve la contesa estraendo un numero pseudo-casuale e confrontandolo con una **probabilità di interdizione di successo (P_{int})**.

La formula che governa questo valore critico unisce le statistiche contrapposte dei giocatori coinvolti:

$$P_{int} = \frac{AttackPower_{dif}}{AttackPower_{dif} + DefensePower_{medio}} \tag{4.3}$$

in cui la resistenza media del passaggio è calcolata come media aritmetica della forza fisica e del controllo palla tra colui che effettua il lancio e colui che lo riceve:

$$DefensePower_{medio} = \frac{DefensePower_u + DefensePower_v}{2} \tag{4.4}$$

Questa equazione fratta, ampiamente diffusa nelle simulazioni agonistiche RPG, possiede proprietà analitiche fondamentali che impediscono rotture meccaniche del gioco (il cosiddetto *power creep*):

- Se i due parametri si equivalgono ($AttackPower_{dif} = DefensePower_{medio}$), il rapporto si riduce esattamente a $\frac{1}{2}$, offrendo una probabilità di vittoria bilanciata al 50% e lasciando l'esito puramente al caso.
- **Asintoto per strapotere difensivo:** Qualora il difensore sia un fuoriclasse rispetto a due passatori modesti ($AttackPower_{dif} \to \infty$), il denominatore viene dominato dal termine dell'attacco, facendo convergere il limite di P_{int} asintoticamente a 1.0. L'intercettazione diventa quasi certa, ma non è mai matematicamente garantita.
- **Asintoto per strapotere offensivo:** Al contrario, se la coppia offensiva è inarrestabile e il difensore debole ($DefensePower_{medio} \to \infty$), la frazione tende asintoticamente a 0, simulando l'impossibilità di scalfire una fitta rete di possesso fisico (il pallone circola al sicuro).

4.1.3 Modalità di Combattimento: Distruttiva vs Riduttiva

Per garantire modularità, il GameManager espone parametri di configurazione che alterano drasticamente il modo in cui il grafo subisce i danni dall'interdizione:

1. **Modalità Distruttiva (*Destructive*):** Questo approccio estremo modella una spazzata o una rottura totale del gioco. Se il check probabilistico fallisce a svantaggio dell'attaccante, l'arco bersaglio viene rimosso dal grafo:

$$c(u, v) = 0 \implies \text{Rimozione Arco} \tag{4.5}$$

   Questa modalità punisce severamente le reti rade e i lanci lunghi (che costituiscono singoli colli di bottiglia), favorendo moduli di gioco ultra-compatti.

2. **Modalità Riduttiva (*Reductive*):** Simula invece l'attrito e lo sporcamento delle traiettorie. Il successo del pressing non cancella la linea di passaggio, ma

ne sottrae una quota unitaria di flusso:

$$c(u, v) \leftarrow c(u, v) - 1 \tag{4.6}$$

Un arco viene rimosso solo e soltanto se il suo logoramento continuo lo porta a una capacità $c(u, v) < 1.0$. Questa opzione incentiva i passaggi ad altissima capacità $c(u, v)$ iniziali, i quali possono sopravvivere a molteplici incursioni avversarie prima di spezzarsi definitivamente.

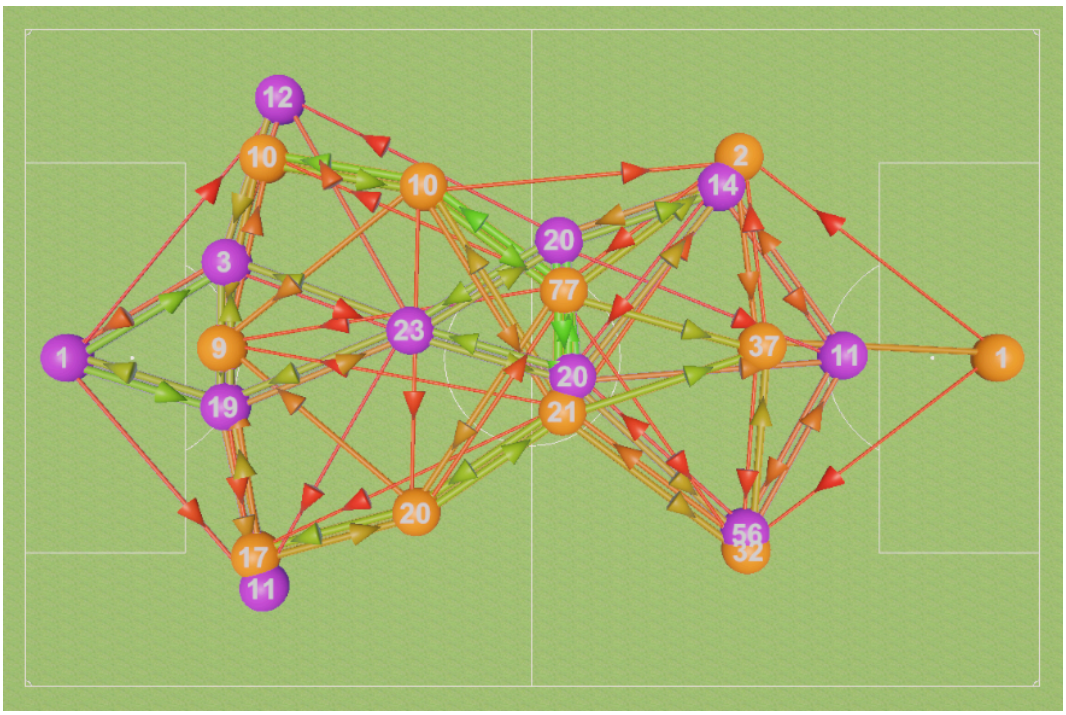

Figura 4.2: Visuale tattica della fase di Combat: si notino le sovrapposizioni tra i dischi di pressing e le linee di passaggio.

4.2 Valutazione del Grafo Residuo

Al termine della fase di Network Interdiction, il motore di calcolo deve quantificare analiticamente lo stato finale delle due formazioni. Le due classi metriche che sintetizzano l'efficacia tattica sono la potenza penetrativa offensiva (misurata tramite il Flusso Massimo) e l'impermeabilità difensiva (valutata tramite la Clique Massima). Tali metriche sono risolte invocando ciclicamente la classe statica 'GraphAnalyzer.cs'.

4.2.1 Instradamento Offensivo: Implementazione C# di Edmonds-Karp

Per la valutazione offensiva, il sistema invoca l'algoritmo esatto di **Edmonds-Karp** (introdotto formalmente nella Sezione 2.1.4). In input, il modulo riceve la matrice di adiacenza asimmetrica rappresentante gli archi superstiti post-combat.

```
public static FlowResult CalculateMaxFlow(float[,] adjacencyMatrix,
    int sourceIndex, int sinkIndex)
{
    int size = adjacencyMatrix.GetLength(0);
    FlowResult result = new FlowResult();
    result.flowMatrix = new int[size, size];

    if (sourceIndex < 0 || sourceIndex >= size ||
        sinkIndex < 0 || sinkIndex >= size) return result;

    // 1. Inizializzazione Rete Residua e Copia Capacità
    int[,] residualGraph = new int[size, size];
    int[,] initialCapacity = new int[size, size];

    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            int cap = Mathf.RoundToInt(adjacencyMatrix[i, j]);
            residualGraph[i, j] = cap;
            initialCapacity[i, j] = cap;
        }
    }

    int maxFlow = 0;
    int[] parent = new int[size];

    // 2. Ciclo di ricerca dei cammini aumentanti tramite BFS
    while (BFS(residualGraph, sourceIndex, sinkIndex, parent, size))
    {
        int pathFlow = int.MaxValue;

        // Identificazione del collo di bottiglia (Min-Cut locale del cammino)
        for (int v = sinkIndex; v != sourceIndex; v = parent[v])
        {
            int u = parent[v];
            pathFlow = Mathf.Min(pathFlow, residualGraph[u, v]);
        }

        // Aggiornamento della rete residua
        for (int v = sinkIndex; v != sourceIndex; v = parent[v])
        {
```

```
            int u = parent[v];
            residualGraph[u, v] -= pathFlow;
            residualGraph[v, u] += pathFlow;
        }

        maxFlow += pathFlow;
    }

    // 3. Estrazione dei flussi per la visualizzazione a schermo
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            if (initialCapacity[i, j] > 0)
            {
                int flow = initialCapacity[i, j] - residualGraph[i, j];
                result.flowMatrix[i, j] = Mathf.Max(0, flow);
            }
        }
    }

    result.totalFlow = maxFlow;
    return result;
}
```

Stabilità Numerica e Discretizzazione

A livello ingegneristico, è di vitale importanza notare la presenza della funzione 'Mathf.RoundToInt(adjacencyMatrix[i, j])' alla linea 20 del frammento di codice soprastante. I dati geometrici all'interno di Unity Engine sono conservati come numeri a virgola mobile ('float'). Qualora Edmonds-Karp dovesse operare su matrici residue floating-point, la precisione limitata dei calcolatori (i cosiddetti *floating-point errors*) porterebbe al progressivo accumularsi di tolleranze vicine allo 0 (es. 10^{-7}).

Questo fenomeno genererebbe due aberrazioni letali per il software:

1. **Flussi Spuri (*Ghost Flows*):** Il ciclo 'while(BFS(...))' individuerebbe cammini la cui capacità dovrebbe fisicamente essere chiusa a zero, restituendo percorsi di luce a video del tutto irrilevanti e fuorvianti per il giocatore.
2. **Non Convergenza (Stallo):** Come dimostrato originariamente da Ford e Fulkerson nel 1956, in presenza di capacità irrazionali l'algoritmo non garantisce la convergenza, causando un blocco del *main thread* (il gioco andrebbe in freeze, distruggendo il framerate). Arrotondare le portate ad interi garantisce che l'incremento

di flusso avvenga solo a passi discreti ≥ 1, blindando la terminazione dell'algoritmo ed azzerando la vulnerabilità a micro-residui infinitesimi.

4.2.2 Misurazione della Coesione tramite la Clique Massima

Al fine di determinare una grandezza commensurabile da poter scontrare algebricamente contro il Flusso Massimo, l'assetto difensivo della squadra non in possesso di palla viene analizzato cercando la sua **Clique Massima**.

Nella Teoria dei Grafi, una cricca (clique) all'interno di una rete è un sottografo i cui nodi sono tutti direttamente interconnessi a due a due. Sotto l'astrazione calcistica, una clique difensiva rappresenta un raggruppamento di atleti che stazionano ad una distanza reciproca sufficientemente breve da potersi scambiare coperture in modo istantaneo e bidirezionale, chiudendo tutti gli spazi.

La natura strutturale della cricca presuppone la bilateralità delle connessioni; per questo motivo, l'algoritmo di estrazione della Clique deve innanzitutto isolare le porzioni non orientate del grafo originale asimmetrico:

$$(u, v) \in E_{simm} \iff c(u, v) > 1.0 \quad \text{AND} \quad c(v, u) > 1.0 \tag{4.7}$$

Rete di Flusso (Asimmetrica) Clique Identificata (Dim. 3)

Figura 4.3: Processo di conversione dalla rete direzionale asimmetrica al grafo simmetrico non orientato. Il nodo D viene scartato poiché manca la connessione reciproca $D \to B$. Il risultato evidenzia la Clique Massima (A, B, C) di dimensione $\omega(G) = 3$.

Il codice C# che esegue tale identificazione si affida alla versione con ottimizzazione *Pruning* del classico metodo di backtracking per enumeratione (una variante *branch-and-bound* ispirata all'approccio Bron-Kerbosch analizzato nel Capitolo 2):

```
public static int CalculateMaxCliqueSize(float[,] adjacencyMatrix)
{
    int n = adjacencyMatrix.GetLength(0);
    if (n == 0) return 0;

    // 1. Estrazione del Grafo Non Orientato Simmetrico
    bool[,] adj = new bool[n, n];
```

```
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            adj[i, j] = adjacencyMatrix[i, j] > 1.0f;
        }
    }

    int maxClique = 0;
    List<int> currentClique = new List<int>();
    List<int> candidates = new List<int>();
    for (int i = 0; i < n; i++) candidates.Add(i);

    FindCliques(adj, currentClique, candidates, ref maxClique);
    return maxClique;
}

private static void FindCliques(bool[,] adj, List<int> current,
    List<int> candidates, ref int maxClique)
{
    if (candidates.Count == 0)
    {
        if (current.Count > maxClique) maxClique = current.Count;
        return;
    }

    // 2. OTTIMIZZAZIONE BRANCH AND BOUND (PRUNING)
    if (current.Count + candidates.Count <= maxClique) return;

    while (candidates.Count > 0)
    {
        int v = candidates[0];
        candidates.RemoveAt(0);

        List<int> nextCandidates = new List<int>();
        foreach (int u in candidates)
        {
            if (adj[v, u] && adj[u, v]) // Requisito di Bilateralità Assoluta
            {
                nextCandidates.Add(u);
            }
        }
```

```
            current.Add(v);
            FindCliques(adj, current, nextCandidates, ref maxClique);
            current.RemoveAt(current.Count - 1);
        }
}
```

Necessità della Potatura (Pruning)

Come accennato nella teoria, la ricerca della Clique Massima è un problema NP-completo; il suo tempo di esecuzione cresce esponenzialmente al variare del numero dei nodi. Sebbene il numero massimo di atleti (11) mantenga il problema in dimensioni contenute, la chiamata frequente e real-time ad una funzione $O(2^n)$ scatenerebbe fastidiosi micro-scatti (*stuttering*) compromettendo la latenza del gioco.

La linea 36 del blocco implementativo:

```
if (current.Count + candidates.Count <= maxClique) return;
```

rappresenta un esempio di **Pruning di Cardinalità**. L'algoritmo controlla proattivamente se, persino nell'ipotesi estrema in cui tutti i restanti candidati disponibili potessero fondersi col ramo attuale, si riuscirebbe comunque ad eguagliare o superare la miglior clique temporaneamente salvata in 'maxClique'. Qualora la risposta fosse negativa, il ramo è considerato matematicamente sterile ed il motore interrompe l'esplorazione profonda, scartando l'intero albero di derivazione. Tale filtro si rivela di fondamentale efficacia ingegneristica per preservare le prestazioni assolute del software.

Capitolo 5

Risultati e Sviluppi Futuri

In questo capitolo vengono analizzati in dettaglio i risultati quantitativi e qualitativi ottenuti dalle sessioni di gioco e di simulazione condotte sul prototipo di **OptiSoccer**. Viene esaminata la risposta sistemica del motore algoritmico a differenti disposizioni tattiche, valutando la sensibilità delle euristiche introdotte rispetto al variare delle formazioni sul campo. Oltre all'analisi empirica degli scenari PvP e dei casi limite che portano a degenerazione topologica, la trattazione si chiuderà presentando gli orizzonti di ricerca futuri, focalizzandosi sull'automazione della fase difensiva (tramite IA algoritmica basata su Min-Cut) e sull'introduzione di complesse dinamiche di rete in tempo reale.

5.1 Test di Gioco e Analisi Numerica

La validazione empirica del modello matematico di *OptiSoccer* è stata condotta attraverso decine di partite di collaudo in modalità giocatore contro giocatore (PvP) locale. Per valutare la correttezza algoritmica, l'osservazione si è concentrata sulla reazione del sistema a due stili tattici calcistici storicamente antitetici, per verificare se il flusso calcolato rispecchiasse l'effettiva robustezza del modulo schierato.

5.1.1 Scenario A: Possesso e Densità (Tiki-Taka)

Il primo test ha previsto lo schieramento di una formazione atta al controllo del pallone (modulo 4-3-3 o 4-2-3-1 a baricentro alto). I giocatori selezionati nel draft prediligevano statistiche di $PassRange$ e $DefensePower$, disposti a distanze ravvicinate all'interno delle loro ristrette *Affinity Zones*.

Dal punto di vista topologico, questa disposizione ha generato un grafo dei passaggi iper-denso, caratterizzato da un'elevata **connettività per archi** e massiccia ridondanza dei cammini aumentanti. In fase di *Combat*, nonostante il pressing avversario sia riuscito a degradare o distruggere con successo quasi il 30% degli archi generati, la topologia della rete ha permesso all'algoritmo di Edmonds-Karp di instradare agilmente il flusso su direttrici secondarie non interdette.

Come illustrato nella Tabella 5.1, il Flusso Massimo superstite post-combat (Φ) si è mantenuto elevatissimo, garantendo una transizione offensiva schiacciante, a

scapito però di un'ingente spesa in crediti in fase di Draft per i centrocampisti di palleggio.

5.1.2 Scenario B: Contropiede Verticale (Lancio Lungo)

In netto contrasto, il secondo test ha esaminato un modulo difensivo votato al contropiede (5-3-2), schierato con baricentro estremamente basso a ridosso del proprio portiere, volto a cercare la punta isolata tramite passaggi a lungo raggio scavalca-centrocampo.

La rete generata è risultata intrinsecamente rada. Sebbene la *Clique Massima* difensiva registrata fosse notevole (data la prossimità dei 5 difensori), l'assenza di nodi intermedi ha lasciato tutto il peso offensivo su un paio di lunghi archi orientati. Questa topologia si è rivelata catastroficamente **vulnerabile all'interdizione**. Le poche direttrici disponibili rappresentavano esse stesse l'intero *taglio minimo* della rete; è bastato che i difensori avversari concentrassero il pressing su quei due o tre archi nevralgici per distruggerli, disconnettendo completamente il pozzo dalla sorgente e facendo crollare il flusso massimo istantaneamente a 0.

Tabella 5.1: Confronto numerico medio tra i due scenari tattici post-combat

Scenario Tattico	**Archi Iniziali**	**Archi Sopravvissuti**	**Flusso Massimo (Φ)**	**Clique Difesa (ω)**
Tiki-Taka (4-3-3)	42	29 (69%)	18.0	2
Contropiede (5-3-2)	18	11 (61%)	0.0	5

5.1.3 Analisi dei Casi Limite (Formazioni Estreme)

Durante il collaudo, per verificare la tenuta matematica dell'Affinity Multiplier e dei vincoli di grafo, sono stati forzati degli scenari assurdi dal punto di vista calcistico ma critici a livello algoritmico:

- **La Formazione All-Defensive (Catenaccio Estremo):** Schierando i 10 giocatori di movimento in prossimità del proprio portiere, l'algoritmo di Bron-Kerbosch ha registrato un valore eccezionale di **clique massima difensiva** $\omega(G) = 8$. L'ammassamento ha generato un blocco impenetrabile, ma ha isolato totalmente la metà campo avversaria, riducendo Φ a zero. L'esito tipico per chi affronta questa squadra è uno 0-0 matematico, simulando la tattica del "parcheggiare l'autobus".
- **La Formazione All-Offensive (Attacco Totale):** Spostando difensori e mediani direttamente nell'area avversaria, il sistema ha calcolato per ciascuno di essi una mostruosa penalità di *Affinity Multiplier* (come descritto nella Sezione 3.4). Il raggio efficace R_{eff} dei difensori è sceso al 10% del nominale, frammentando la rete in nodi isolati. Senza i raggi necessari per creare archi, la sorgente è rimasta scollegata e il flusso è crollato. Questa validazione certifica l'efficacia del decadimento cubico nel disincentivare le formazioni sbilanciate.

5.2 Score-Matching e Applicazione Strategica del Taglio Minimo

Le sessioni PvP hanno dimostrato un rapido sviluppo delle abilità cognitive dei giocatori. Nel corso dei match, gli utenti smettono di posizionare i difensori per pressare i nodi casualmente, ma imparano ad eseguire intuitivamente il riconoscimento visivo del **Taglio Minimo (Min-Cut)**.

Il Teorema Max-Flow Min-Cut di Ford-Fulkerson (Sezione 2.1.3) sancisce che la capacità del collo di bottiglia eguaglia il flusso massimo. I giocatori esperti hanno applicato attivamente questo teorema: identificando gli archi critici del taglio minimo nemico e riversando su di essi le loro *Aree di Pressing* (grazie alla logica di priorità descritta nella Sezione 4.1), sono riusciti a massimizzare l'interdizione della rete avversaria spendendo il minimo numero di difensori necessari, un brillante esempio di ottimizzazione indotta dall'interfaccia visiva.

5.2.1 Sensibilità dell'Equazione di Bilanciamento

L'integrazione tra la fase offensiva e quella difensiva avviene nel calcolo dei Gol, dove la potenza di calcolo si sintetizza nella formula di *Score-Matching*:

$$Gol_A = \max\left(0, \lfloor \alpha \cdot \Phi_A - \beta \cdot \omega(B) \rfloor\right) \tag{5.1}$$

Nei playtest iniziali ($\alpha = 1, \beta = 1$), si è notato che la clique difensiva avversaria, per quanto compatta, difficilmente superava valori di 4 o 5, mentre un ottimo flusso poteva raggiungere valori pari a 15. Per bilanciare il gioco evitando punteggi tennistici irreali (es. 10 a 2), i coefficienti di calibrazione sono stati aggiustati (ad esempio $\alpha = 0.3$), comprimendo il divario e riportando i risultati a distribuzioni realistiche (2-1, 1-0, 0-0). Questa formula garantisce che un'ottima disposizione difensiva ($\omega(B) \geq 4$) basti ad azzerare le velleità di attacchi frammentati o deboli.

5.3 Sviluppi Futuri della Ricerca

Il completamento del framework architetturale di *OptiSoccer* apre un ventaglio di esplorazioni future, tanto nell'ambito dell'Intelligenza Artificiale quanto nell'evoluzione dei modelli di Network Interdiction interattivi.

5.3.1 Intelligenza Artificiale Difensiva basata su Algoritmi Esatti (PvAI)

Un limite dell'attuale versione è la necessità di due giocatori umani. Lo sviluppo più promettente è la realizzazione di una IA tattica (PvAI) in grado di schierare autonomamente la squadra difensiva. Tale AI non si affiderebbe a reti neurali *black-box*, ma poggerebbe le basi direttamente sui teoremi di ottimizzazione lineare:

1. L'IA genererebbe il grafo dei passaggi potenziale della squadra umana.

2. Applicherebbe internamente Edmonds-Karp per individuare esattamente gli archi che costituiscono il taglio minimo della rete nemica.

3. Un risolutore di ottimizzazione posizionale continuo (come un campo di forze potenziale, in cui il taglio agisce da attrattore gravitazionale) posizionerebbe le coordinate spaziali dei difensori della CPU in modo da sovrapporre matematicamente i loro dischi di pressing agli archi del taglio minimo.

Questo approccio creerebbe un'IA "matematicamente perfetta", spingendo l'umano a dover creare reti altamente ridondanti per superare un difensore guidato dai teoremi stessi della Ricerca Operativa.

5.3.2 Network Fortification e Jamming

Per espandere la complessità strategica della fase offensiva, il futuro del modello prevede l'introduzione di meccaniche di **Fortificazione (Network Fortification)**.

Come previsto nei modelli avanzati di Interdiction Games, l'attaccante potrebbe impiegare punti risorsa limitati per blindare determinati archi strategici della propria rete di passaggi, riducendone la vulnerabilità al pressing.

$$P_{int\_fortificato} = P_{int} \cdot (1 - \gamma), \quad \text{con } \gamma \in (0, 1) \tag{5.2}$$

Questa variazione parametrica imporrebbe al giocatore in fase offensiva un'ulteriore sfida combinatoria: capire su quali colli di bottiglia investire il proprio budget difensivo per massimizzare la probabilità di sopravvivenza del flusso.

5.3.3 Dall'Analisi Discreta a Grafi Fluidi in Tempo Reale

L'orizzonte tecnologico ultimo prevede il superamento della rigida turnazione a fasi statiche in favore di un sistema a **grafi fluidi e continui** basato su modelli a forze (*Force-Directed Graphs*).

I calciatori, anziché essere posizionati manualmente a turno, verrebbero gestiti come entità semi-autonome spinte da vettori di attrazione (verso le zone di affinità o il portatore di palla) e vettori di repulsione (allontanamento dai pressatori). Edmonds-Karp e Bron-Kerbosch verrebbero ricalcolati in background in maniera asincrona ogni pochi millisecondi. Il giocatore agirebbe come un "regista" che imposta le macro-tattiche, osservando a schermo il grafo pulsare, cambiare conformazione, e le luci dei flussi spostarsi dinamicamente in risposta al mutare degli schieramenti, offrendo la simulazione calcistica algoritmica definitiva.

Capitolo 6

Conclusioni

In questo capitolo finale vengono riassunti in modo organico gli obiettivi raggiunti e formulate le considerazioni conclusive sull'intero ciclo di progettazione, modellazione matematica e sviluppo software che ha dato luce al *serious game* **OptiSoccer**. Verrà inoltre analizzato l'impatto potenziale di questa specifica metodologia nell'efficacia dell'insegnamento universitario di discipline astratte quali l'Ottimizzazione Combinatoria e la Ricerca Operativa.

6.1 Sintesi del Lavoro Svolto

L'elaborato ha dimostrato concretamente come problemi teorici ed algoritmici considerati ostici nella teoria dei grafi possano essere incapsulati, ingegnerizzati e risolti in tempo reale all'interno di un'esperienza ludica bidimensionale interattiva ed accattivante. La trasposizione logica dall'alveo puramente accademico a quello dell'intrattenimento digitale è stata validata positivamente ad ogni livello.

Attraverso lo sviluppo completo di *OptiSoccer* sul motore grafico *Unity*, sono stati ottenuti i seguenti risultati tecnologici:

- **Modellazione Rigorosa di Reti Dinamiche:** Si è attuata la conversione delle dinamiche calcistiche in un rigoroso formalismo matematico. I calciatori si comportano come nodi vincolati e i passaggi come archi capacitivi. È stata progettata e implementata l'equazione originale del *Decadimento Cubico (Affinity Multiplier)*, che traduce la disciplina tattica spaziale in una dura penalità topologica, validando l'assunto per cui la geometria dello schieramento determina le performance teoriche del network.
- **Decentralizzazione dell'Interdizione di Rete:** È stato implementato un sistema di *Combat* simultaneo che simula i complessi problemi di *Network Interdiction* bi-livello non affidandosi ad un solutore globale centralizzato, ma distribuendo la logica probabilistica di intercettazione direttamente sui singoli agenti fisici (i difensori sul campo). L'analisi dei limiti asintotici di tali equazioni di intercettazione ha dimostrato l'assenza di singolarità matematiche e di degenerazioni ludiche (*power creep*).

- **Valutazione Algoritmica ad Alte Prestazioni:** L'integrazione di celebri algoritmi all'interno del ristretto ciclo macchina (*Main Loop*) di un videogioco a 60 FPS ha richiesto particolari accortezze ingegneristiche in C#. Si è dimostrato come l'arrotondamento intero metta al riparo l'algoritmo esatto di **Edmonds-Karp** dai bug di precisione della virgola mobile, garantendo l'assenza di loop infiniti e flussi spuri. Parimenti, l'integrazione di rigidi vincoli di *pruning di cardinalità* (*branch-and-bound*) all'interno dell'algoritmo di **Bron-Kerbosch** ha scongiurato l'esplosione esponenziale dei tempi di ricerca per la Clique Massima.

- **Equazione di Bilanciamento (Score-Matching):** L'unione delle metriche di Flusso Massimo (attacco) e Clique Massima (difesa) all'interno di una formula di punteggio algebricamente bilanciata ha confermato come la tattica calcistica umana e la pura combinatoria matematica convergano al medesimo risultato logico.

6.2 Il Valore Formativo del Mascheramento Algoritmico

L'aspetto più dirompente del progetto risiede nel suo immenso potenziale come **Serious Game** didattico. Le materie afferenti all'Ingegneria Informatica e alla Matematica Applicata (come la Ricerca Operativa) soffrono sovente di un forte "scollegamento tattile". Lo studente manipola matrici e studia dimostrazioni su carta, perdendo tuttavia la percezione geometrica della rete nella sua interezza.

OptiSoccer aggira questa barriera cognitiva avvalendosi della metodologia del *Mascheramento Algoritmico*:

- Il **Flusso Massimo e i Cammini Aumentanti** non sono più incomprensibili array in console, ma si palesano come tubi cilindrici tridimensionali che si ingrossano ed emettono luce in misura proporzionale allo sforzo offensivo transitante.

- L'importantissimo **Teorema del Max-Flow Min-Cut** diventa una sfida tattica naturale: le sessioni di collaudo hanno dimostrato che il giocatore, pur ignorando la formulazione matematica del taglio minimo, ne deduce visivamente l'esistenza, scagliando le proprie difese contro gli archi "collo di bottiglia" che mantengono in vita il gioco avversario.

- Il calcolo delle **Componenti Fortemente Connesse (Clique)** diviene l'inconfutabile misurazione visiva di quanto la propria linea difensiva sia schierata in maniera compatta e mutua, esaltando il valore della cooperazione matematica tra i nodi.

6.3 Considerazioni Finali

In chiusura, questa tesi dimostra che il confine tra l'intrattenimento videoludico e la simulazione scientifica avanzata è straordinariamente sottile. Il connubio interdisciplinare esplorato in questo lavoro apre prospettive affascinanti: lo sviluppo di software *game-based* non sminuisce il rigore ingegneristico, ma al contrario esige un'enorme perizia tecnica per ottimizzare algoritmi sofisticati garantendo un'interazione impeccabile. OptiSoccer getta le basi per una didattica moderna, in cui le equazioni di ottimizzazione non vengono semplicemente lette e memorizzate, ma vengono vissute e "giocate" in tempo reale.

Bibliografia

[1] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[2] Coenraad Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[3] R. Kevin Wood. Triangular network-interdiction problem. *European Journal of Operational Research*, 66(1):97–103, 1993.

[4] Clark C. Abt. *Serious Games.* Viking Press, 1970.

[5] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[6] Richard Wollmer. Removing arcs from a network. *Operations Research*, 12(6):934–940, 1964.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 3rd edition, 2009.

[8] John Adrian Bondy and U. S. R. Murty. *Graph Theory with Applications.* Macmillan London, 1976.

[9] Marc Prensky. *Digital Game-Based Learning.* McGraw-Hill, 2001.