2. Modelado de partidos de fútbol

Para modelar la estrategia de juego de un partido de fútbol, se puede usar un esquema como el mostrado en la figura 1, en donde los 11 jugadores del equipo corresponden a los círculos morados (el número corresponde a su camiseta) y las líneas verdes representan los posibles pases que puede hacer en la situación de juego dada.

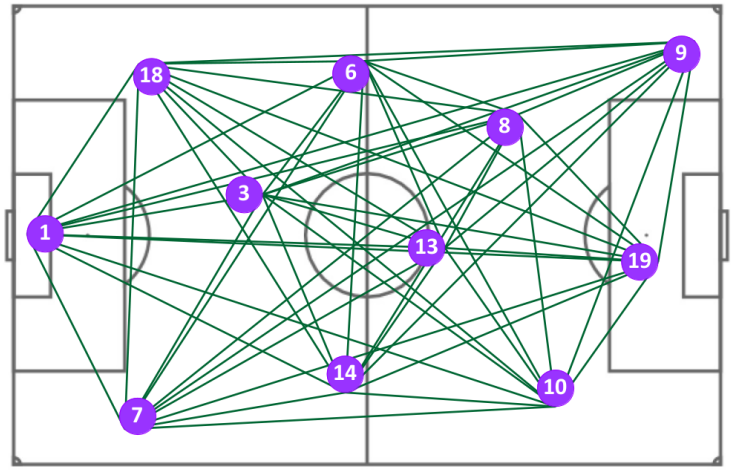

Figura 1: Situación de un instante en un juego de fútbol. Los círculos morados representan los jugadores, mientras que las líneas verdes representan los posibles pases de balón.

Para poder estimar la mejor estrategia de juego del equipo, cada jugador se caracteriza con un conjunto de datos que integra su posición actual en la cancha y su desempeño en jugadas de ataque y de defensa. Cabe anotar que estos desempeños se representan con valores reales contenidos en el rango $[0, 1]$, donde 0 representa un desempeño nulo y 1 representa el máximo desempeño. Además, este valor de desempeño depende, entre otros, del biotipo del jugador, de su cansancio, de su motivación, de la temperatura del terreno, de la cantidad de tarjetas acumuladas, etcétera; es decir, es un valor dinámico.

Ahora, para completar el modelado de la estrategia de juego, cada pase del balón entre dos jugadores debe etiquetarse (o pesarse) con un valor que indique la efectividad del pase. Esta efectividad es inversamente proporcional a la distancia euclidiana entre ambos jugadores (es decir, a menor distancia entre los jugadores el pase es más efectivo) y se calcula como:

$$E\left(a,b\right) = 1 - \frac{\sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}}{\Delta} \tag{1}$$

donde a y b representan los identificadores de los jugadores, $\Delta \in \mathbb{R}^+$ corresponde al largo (en metros) de la diagonal de la cancha de fútbol (usualmente es $133m$), y (a_x, a_y) y (b_x, b_y) corresponden a la posición en el terreno de los jugadores a y b, respectivamente. Note que la efectividad del pase $E\left(a,b\right)$ tiene valores en el rango $[0, 1]$, donde 0 representa un "pésimo" pase y 1 representa un pase "perfecto" de balón entre los jugadores; todo valor intermedio permite la posibilidad de que el pase sea errado o interceptado por un contrario.

El objetivo de esta estrategia de modelado es identificar, en una situación de juego, la mejor jugada que permita realizar un ataque efectivo (jugada con mayor posibilidad de gol) o una defensa efectiva (jugada con mayor posibilidad de reagrupamiento para un nuevo ataque). Para esto, se empieza con el jugador que conduce el balón y, teniendo en cuenta el tipo de jugada que se desea generar (ataque o defensa), se utilizan los porcentajes de desempeño de cada uno de los jugadores y la efectividad de los pases entre ellos para identificar:

- el jugador con mayor posibilidad de realizar bien la jugada (mejor porcentaje de desempeño en ataque o defensa) y
- la secuencia de pases a realizar que permitan llevar la pelota desde el jugador activo al mejor jugador identificado.

En la figura 2 se describe gráficamente el resultado esperado para realizar la *mejor jugada de ataque*. El jugador activo se identifica ahora con el círculo azul. Cada jugador tiene un porcentaje de efectividad en jugadas de ataque,

representado por los números azules, que permiten escoger el jugador con mejor porcentaje de efectividad (en este caso el número 19) y, así calcular la secuencia de pases que llevan de la mejor forma el balón desde el jugador activo al jugador 19. Para esto, el proceso debe tener en cuenta la efectividad de cada pase entre jugadores, y así escoger la ruta que maximice esta efectividad. Adicionalmente, debe tener en cuenta que el pase siempre se realiza en una única dirección, es decir, si el jugador a pasa el balón al jugador b, éste no puede pasarlo de vuelta a a.

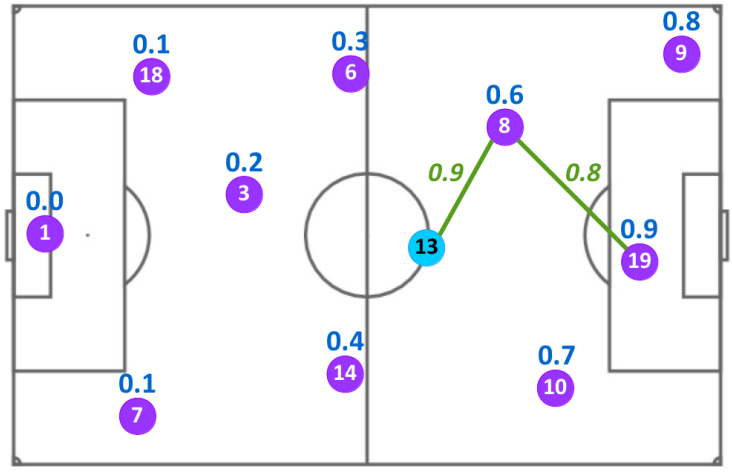

Figura 2: Ilustración del resultado esperado para una jugada de ataque. Desde el jugador activo (azul), hasta el jugador con mejor efectividad en ataque (19), se calcula la secuencia de pases que llevan el balón entre ellos con mayor efectividad. Note que la probabilidad de lograr que esta jugada (la mejor en ataque en esta situación) se realice es de 0,72.

De forma similar, la figura 3 describe el resultado esperado para realizar la *mejor jugada de defensa*. En este caso, los porcentajes de efectividad en jugadas de defensa para cada jugador se representan con los números rojos. En este caso, el jugador con mejor desempeño es el número 7, y la secuencia de pases que llevan el balón entre el jugador activo y el jugador 7, de la forma más efectiva posible, está representado con líneas verdes.

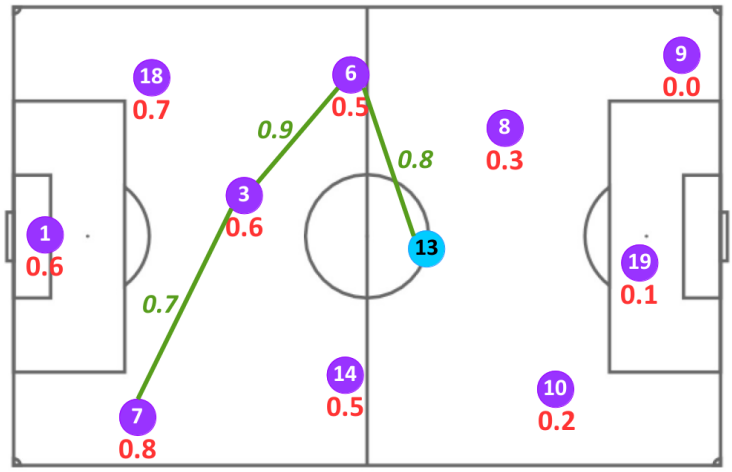

Figura 3: Ilustración del resultado esperado para una jugada de defensa. Desde el jugador activo (azul), hasta el jugador con mejor efectividad en defensa (7), se calcula la secuencia de pases que llevan el balón entre ellos con mayor efectividad. Note que la probabilidad de lograr que esta jugada (la mejor en defensa en esta situación) se realice es de 0,504.

Usted debe implementar los métodos asociados a:

- Dados los identificadores de dos jugadores en la cancha, calcula la efectividad del pase del balón entre los dos, teniendo en cuenta la ecuación 1.

- Dada una situación de juego y el identificador del jugador activo (quien tiene el balón), generar todas las secuencias de pases óptimas para todos los demás jugadores. Tenga en cuenta que la efectividad final de los pases implica multiplicar los pesos en vez de sumarlos y que, además, las efectividades más altas tienen mayor prioridad (es decir, el costo de un pase es $C(a,b) = 1 - E(a-b)$).
- Dada una situación de juego, descrita por el identificador del jugador activo (quien tiene el balón) y una estrategia (ataque o defensa), identifica el jugador con mejor desempeño en la estrategia, encuentra la ruta de mayor efectividad de pase entre el jugador actual y el mejor jugador identificado, la probabilidad final de la jugada y, finalmente, informa si esa jugada involucró a todos los jugadores en la cancha.

2.1. (60 %) Algoritmos

Escriba la implementación de los siguientes algoritmos en C++:

1. (15 %) Calcular efectividad del pase entre dos jugadores.
2. (20 %) Obtener todas las secuencias de pases óptimas.
3. (25 %) Informar la mejor secuencia de juego.

Las implementaciones deberán tener en cuenta las siguientes clases en C++:

```
class Jugador
{
protected:
  int m_Numero; // *
  float m_PosX; // *
  float m_PosY; // *
  float m_TamanoTerreno; // *

public:
  Jugador( ); // *
  virtual ~Jugador( ); // *

  float ObtenerDesempenoAtaque( ) const; // *
  float ObtenerDesempenoDefensa( ) const; // *

  float CalcularEfectividadHasta( const Jugador& otro ) const;
};

class SituacionDeJuego
{
protected:
  std::vector< Jugador > m_Jugadores;
  float m_TamanoTerreno; // *

public:
  SituacionDeJuego( ); // *
  virtual ~SituacionDeJuego( ); // *

  struct NodoSecuenciasOptimas
  {
    int ElPaseVieneDesde;
  };
  typedef std::vector< NodoSecuenciasOptimas > SecuenciasOptimas;

  SecuenciasOptimas ObtenerSecuenciasOptimas( int numero, bool esAtaque ) const;

  struct SecuenciaOptima
  {
    std::vector< int > Secuencia;
    float ProbabilidadFinal;
    bool TodosLosJugadores;
  };

  SecuenciaOptima ObtenerSecuenciaOptima( int numero, bool esAtaque ) const;
};
```

Suponga que los métodos comentados al final con "`// *`" ya están implementados (es decir, usted no debe escribirlos). Así mismo, los atributos marcados de la misma manera son accesibles con métodos Set/Get que también ya están implementados.

Para la implementación, tenga en cuenta (será tenido en cuenta en la calificación):

- el no uso de salidas/entradas por pantalla/teclado (i.e. paso/retorno correcto de valores y/o objetos), y
- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL o en las estructuras descritas y comentadas más arriba.

2.1.1. **(15 %)** `float Jugador::CalcularEfectividadHasta( const Jugador& otro ) const`

```
{
}
```

2.1.2. **(20 %)** `SituacionDeJuego::SecuenciasOptimas SituacionDeJuego::ObtenerSecuenciasOptimas( int numero, bool esAtaque ) const`

```
{
}
```

2.1.3. **(25 %)** `SituacionDeJuego::SecuenciaOptima SituacionDeJuego::ObtenerSecuenciaOptima( int numero, bool esAtaque ) const`

```
{

}
```