علی فراشبندی

ربات فوتبالیست :کارایی

.گلزدن: ارسال توپ به دروازه با دقت بالا و توانایی مراعات فاصله و زاویه

دفاع کردن: انجام حرکات دفاعی مناسب مانند انتقال به سریعی به جلو، بلاک کردن شوت حریفان، و جلوگیری از ایجاد فرصت گل .برای حریفان

.پاس دادن: انتقال توپ به همتیمیها با دقت و زمانبندی مناسب .بیرون نرفتن از زمین: حفظ موقعیت در محدوده زمین و جلوگیری از خروج از محدوده مجاز .برد بازی: انجام عملکردهای مناسب تا به نتیجه مطلوب در بازی برسد، مانند نگهداشتن مالکیت توپ و ساخت فرصتهای گل

:مثالهای دیگر .تحلیل موقعیت بازی: تشخیص موقعیت مناسب برای حرکت و انتقال توپ با توجه به موقعیت بازیکنان حریف و همتیمی .تشخیص خطا توسط داور: تشخیص واقعههایی مانند آفساید، خطاها و جریمههای دیگر توسط داوران بازی

:محیط .کادر بیرون بازی: تشخیص موقعیت کادر بیرون بازی و رفتار مناسب در برابر آن، مانند توقف بازی یا ادامه آن .داور بازی: تشخیص حرکات و تصمیمات داوران و انطباق با آنها

:عملگرها .دریبل: حفظ کنترل توپ در حالت حرکت و ایجاد فرصتهای گل برای تیم خود .گرفتن توپ از بازیکن حریف: انجام عملیات تکل و سرقت توپ از بازیکن حریف به منظور کسب مالکیت توپ

:سنسور .تشخیص بازیکن حریف و خودی : تشخیص و مدیریت موقعیت بازیکنان خود و حریفان در زمین .سنسور دوربین ۳۶۰ درجه: استفاده از دوربین برای تشخیص موقعیت دقیق توپ و بازیکنان در تمام زمین .سنسور رد توپ از دروازه: تشخیص و ثبت خروج توپ از دروازه برای اعلام گل .سنسور تشخیص دروازه خودی: تشخیص موقعیت و محدوده دروازه خودی و جلوگیری از ورود توپ به آن

علی فراشبندی

۸ وزیر

-----------------------------------

-

def is\_safe(board, row, col):

   # Check this column on upper side

   for i in range(row):

       if board[i] == col or \

          board[i] - i == col - row or \

          board[i] + i == col + row:

           return False

   return True

def print\_board(board):

   n = len(board)

   for row in range(n):

       line = ""

       for col in range(n):

           if board[row] == col:

               line += "Q "

           else:

               line += ". "

       print(line)

   print("\n")

def solve\_n\_queens(n, row=0, board=None):

   if board is None:

       board = [-1] \* n

   if row == n:

       print\_board(board)

       return True

   for col in range(n):

       if is\_safe(board, row, col):

           board[row] = col

           if solve\_n\_queens(n, row + 1, board):

               return True

           board[row] = -1

   return False

# Run the program with 8 queens

solve\_n\_queens(8)

**تمرین مسائل csp**

همانطور که میدانید مسئله ها میتوانند از طریق جست وجو در فضایی از حالتها حل شوند. این حالتها را میتوان با ابتکارهای خاص دامنه ارزیابی، و تست کرد که آیا آنها حالتهای هدف هستند یا خیر. اما از نظر الگوریتم جست وجو هر حالت یک جعبه سیاه است که ساختار داخلی آن قابل تمیز دادن نیست.

در اینجا روشی داریم که تعداد زیادی از مسئله ها را با کارایی بیشتری حل میکند. برای هر حالت، از مجموعه ای از متغیرها که هر کدام دارای یک مقدار است، استفاده میکنیم. مسئله وقتی حل میشود که هر متغیر دارای مقداری باشد که تمام محدودیت های روی آن متغیر را برآورده کند (آن محدودیت را ارضا کند). مسئله ای که به این روش توصیف میشود، مسئله ارضای محدودیت یا CSP نام دارد.

الگوریتم جست وجوی CSP از ساختار حالت استفاده میکند و به جای ابتکارهای خاص مسئله، از ابتکارهای همه منطوره برای حل مسئله های پیچیده بهره میگیرند. ایده ی اصلی این است که با مشخص کردن ترکیب هایی از متغیر/ مقدار که محدودیت ها را نقض میکنند، بخش های بزرگی از فضای جست وجو بطور همزمان حذف شوند.