تقييم المسار الأفضل بين موقعين

عباس محسن عبد الحسين ميراس سلمان جواد
كلية العلوم-جامعة بابل

الخلاصة

إن إيجاد المسار الأفضل بين موقعين له أهمية كبيرة في تخطيط مسار الإنسان الآلي ضمن بيئة معينة, إيجاد المسار الأمثل بين مدينتين في خارطة مدن ,كذلك إيجاد الطريق الأمثل لأجهزة الشرطة وسيارات الإسعاف الفوري للوصول إلى الهدف[4,2].

تم تقييم المسار الأفضل مابين موقعين باستخدام أحد الخوارزميات التقليدية لهياكل البيانات (Dijksra Algorithm) والتي كانت شائعة الاستخدام لهذا الغرض وإحدى تقنيات الذكاء الاصطناعي ضمن تقنية حل المسائل (Problem solving technique) متضمنة طريقة للبحث لأيجاد المسار الأفضل (Best path search (A* algorithm)) ونتيجة لهذا التقييم ظهر ان التقنية الثانية (A* search technique) هي الأفضل وذلك لأنها توازن مابين المسار الأقصر ووزن العائق للموقع وكذلك تكتفي بحل امثل واحد (One optimal solution).

طرائق البحث (Search Technique)[1,7]:

في اغلب الأحيان نرى ان الكثير يبحث عن الخوارزمية المثلى لايجاد افضل مسار بين موقعين ,حيث ان الوصول الى ايجاد المسار الأمثل غالبا مايكون مكلف جدا. ولغرض اجراء عملية البحث يجب ان يكون لدينا مخطط, وممكن ان تستخدم المخططات في خارطات الطرق (road map) وفي شبكات الأتصالات (communication networks) . ففي خارطة المدن يعبر عن المدن بالعقد (nodes) ويعبر عن الطرق الواصلة بين تلك المدن بخطوط مستقيمة تصل تلك العقد في ذلك المخطط.

ان حل مشكلة ايجاد افضل مسار تعني الحصول على مسار اولي بين العقدتين الذان يمثلان المدينتين المعلومتين وان المسار الأمثل هو المسار الذي تكون جميع النقاط المكونة له مختلفة , أي لايوجد تكرار لاي عقدة ضمن ذلك المسار.

من السهل جدا ايجاد المسار الأقصر بين عقدة المصدر (source node) وعقدة الهدف (goal node) او الى أي عقدة عقدة اخرى في مخطط المدن او أي مخطط اخر بغض النظر عن امثلية ذلك المسار. لغرض البحث في امثلية المسار بين اية عقدتين يجب التعرف على الاتي :

- التنقيب (Heuristic):

ان كلمة (heuristic) هي كلمة اغريقية جاءت من الكلمة (heuriskein) وتعني (to discover) أي طريقة تعيين الهدف المطلوب.

وبالأمكان ان تعرف بمجموعة من التجارب العملية التي تعتمد على مجموعة قوانين لأختيار الحالة اللاحقة (next state) للحالة الحالية (current state) اثناء عملية البحث .

كلما كانت طرق التنقيب (heuristic) جيدة نستطيع الحصول على حل جيد للمشاكل الصعبة , ومن الأمثلة التي يطبق فيها التنقيب (heuristic) بشكل جيد هي مسألة (nearest neighbor) المطبقة على خوارزمية (sales man)والتي تعتبر تطوير لخوارزمية افضل كلفة(best cost search) .

ان طرق التنقيب تكون غير واضحة المعالم ما لم نستخدم دالة التقييم (evaluation function):

- دالة التقييم (Evaluation Function):

ان لكل عقدة هنالك قيمة تمثل كلفة العقدة حيث يكون الأنتقال من الحالة (العقدة) الحالية (current state) الى الحالة(العقدة) التالية (next state) التي تملك اقل قيمة وتسمى ايضا معوقات الحركة من عقدة الى عقدة اخرى، وهكذا يكون الأنتقال في المخطط اعتمادا على هذه القيم التي تمثل كلفة المسارات للوصول الى الهدف (final state)، ولقد تم استخدام هذه الدالة مع كل من خوارزمية البحث العمقي ابتداء (Depth First Search) وخوارزمية البحث العرضي ابتداء (Breadth First Search) للحصول على طرق جديدة اكثر تطور هي:

Hill-Climbing search = Depth-First search + evaluation function.
Best-First search = Breadth-First search + evaluation function.

حيث عند البحث عن مسار ضمن المخطط الشجري (Tree graph) باستخدام طريقة التسلق (hill-climbing) حيث يكون التسلق او الأنتقال من المستوى الحالي (current level) الى المستوى اللاحق (next level) بأستخدام المسار الذي يمتلك اقل قيمة . اما الطريقة الثانية(Best-First) فيتم الأنتقال من المستوى الحالي الى اللاحق باستخدام اقل قيمة للعقدة بعد ذلك يتم مقارنة المستوى الحالي بالمستوى اللاحق فاذا كانت قيمة المقارنة كبيرة فيتم الرجوع الى نفس المستوى الأول ويتم اختيار العقدة الأخت للعقدة الأولى تمتلك اقل قيمة ونستمر بهذه المقارنة حتى وصولنا الى الهدف (goal state).

ولغرض الحصول على طرق بحث اكثر فعالية يتم تضمين دالة الكلفة (Cost function) للطرق السابقة حيث من المتوقع ان تصبح افضل الطرق المستخدمة في المستقبل.

- دالة الكلفة (Cost function):

هي عبارة عن دالة تعطي كل مسار ابتداء من الحالة الأبتدائية (Initial state) قيمة معرفة حيث يكون الأنتقال الى الحالة التالية التي سوف يمتلك المسار فيها اقل قيمة ، وهكذا يتم الأنتقال الى باقي العقد اعتمادا على اقل مجموع لقيم المسارات من الحالة الأبتدائية الى الحالة النهائية.

وعند تطبيق هذه الدالة تم الحصول على الآتي:

Best-Cost search = Breadth-First + Cost function
Best-First search =
(1) Breadth-First + Cost function + Evaluation function
(2) Best-First + Cost function.

ان الطريقة الأولى تسمى ايضا التفرع والأحتساب (Branch and Bound) وهي قريبة من خوارزمية (Dijksra) ، والطريقة الثانية والتي تعتبر من اهم طرق البحث الأفضل في ايجاد الحل الأمثل وبالأمكان ان تعتبر:

Best-Cost + Evaluation function

وتسمى ايضا (A* Technique). حيث ان هذه الطريقة تحاول البحث للحصول على المسار الأمثل وسرعة جيدة مقارنة مع خوارزمية (Dijksra).

من احدى التطبيقات الشائعة لحل مشكلة ايجاد المسار بين عقدتين في مخطط موزون (weighted graph) او غير موزون (digraph) هو تطبيق (Dijksra) لايجاد اقصر او اقل كلف للمسار بين عقدتين في المخطط [4,5].

من خلال العلاقة:

G=(V,E,W) is

$$\sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

تكون W هي مصفوفة تمثل قيمة المسار V_i الى V_{i+1} أي قيمة المسار بين عقدتين والتي تمثل دالة الكلفة (cost function). ويكون من السهل ايجاد المسار الأقصر من عقدة المصدر الى جميع العقد الموجودة في المخطط او عقدة الهدف, وان خوارزمية (Dijksra) تقوم بعملية جمع قيمة المسار بين عقدة البداية الى العقدة الحالية والأستمرار في الجمع حتى الوصول الى العقدة الهدف وباختيار اقل القيم.

يمثل المسار V_i الى V_{i+1} هو اقصر مسار واقل كلفة اذا لم يوجد هنالك مسار اخر بين العقدتين يسلك باقل كلفة من المسار الأول.

Dijksra Algorithm:

```
Let S is the start sate and F is the final state
Perm = [S]
For I ← 1 to maxnode do
        Distance(I)  ← ∞
Current ← S
Distance(current) ← 0
While current <> f do
Begin
 De    distance(current)
 Small dist ← ∞
  For j ← 1 to maxnode do
If j∉perm then
    Begin
        Sum ← de + cost (current,j)
        If sum < distance (j) then
         Begin
            Distance (j) ← sum
            Preced (j) ← current
    End
        If distance(j) < smalldist then
          Begin
            Smalldist ← distance(j)
       K ← j
    End
        End-if
```

```
        Current ← k
        Perm ← perm +[current]
      End-while
    Di ← distance(F)
End-algorithm
```

Best-Path (A^* Algorithm) : المسار الأفضل

المقصود بالمسار الأفضل الذي يوفر امكانية التوازن بين قصر الطريق وقيمة اقل عائق عند كل عقدة وهي من الأمور المهمة جدا عند تخطيط مسار الأنسان الالي ضمن بيئة معينة . من خلال ملاحظة العلاقة التالية[6,5,3]:

G=(C,V,E,W) is

$$W(C_i - V_i,\ C_{i+1}+V_{i+1})\sum_{i=1}^{k}$$

تكون W مصفوفة تمثل قيمة المسار من $C_i - V_i$ الى $C_{i+1}+V_{i+1}$ بين عقدتين . حيث ان (C) تمثل كلفة المسار في المخطط و (V) تمثل قيمة كل عقدة من عقد المخطط (وزن العائق).

ان هذه الخوارزمية لاتبحث عن قيم العقد في المخطط مالم تكون بحاجة لها وهنا نلاحظ بانها تطرق اقل عدد من العقد للوصول الى الهدف وتقوم بالاحتفاظ بقيم المسار حتى الوصول الى الحالة النهائية والتي تمثل الكلفة الكلية للمسار (Total cost) ويتم حسابها كالاتي:

$$S_1 = (C_i - V_i) \quad \ldots\ldots\ldots\ldots\ldots(1)$$

اذا كانت الحالة الحالية (current state) هي نفسها الحالة الأبتدائية (initial state) فان قيم C_i و V_i تساوي (صفر) ويكون الناتج:

$$(0-0) = 0$$

ثم نجد :

$$S_2 = (C_{i+1}+V_{i+1}) \quad \ldots\ldots\ldots\ldots\ldots(2)$$

للحالة القادمة(الجديدة) (next state) . ثم نقوم بعملية الجمع كالاتي :

$$Total\ cost = S_1+S_2 \quad \ldots\ldots(3)$$

حيث ان S_1 يمثل كلفة المسار الأقصرمن عقدة البداية الى العقدة n في المعالجة و S_2 تعيد الكلفة الفعلية للمسار الأقصر من العقدة n الى الهدف وان الكلفة النهائية (total cost) تمثل الكلفة الفعلية للمسار الأمثل من عقدة البداية ولغاية عقدة الهدف.

A^* Algorithm

```
Let S is the start state and F is the final state
Perm = [S]
For i ← 1 to maxnode do
  distance(i ) ← ∞
current ← S
j ← current
```

```
Cost(current,j )← 0
Value(current,j) ← 0
distance(current )← 0
While current <> F do
      Begin
       Smalldist ← ∞
      de ← distance(current)
      maxev ← ∞
      If current = S then
        Begin
          J← current
          d1← de-value(current,j)
      End
If current <> S then
      Begin
       j← current
       d1 ← de-value(current,j)
      End
For j ← 1 to maxnode do
  Begin
     If not(j in perm) then
     Begin
        sum ← d1+cost(current,j) + value(current,j)
         If sum <= distance(j) then
          Begin
              distance(j )← sum
            Preced(j)← current
           End
If (distance (j) < smalldist) and
(value(current,j) < maxev ) then
begin
   smalldist ← distance(j)
    maxve ← value(current,j);
    k ← j
end
end-if
if(k=current) then
write(The Path is not Found)
current ← k
perm ← perm+[current]
      end-while
      k ← f
       b ← 1,   i ← 0
      while k<>s do
          begin
             temp[p]← k
```

```
                k ← preced(k)
                b=b+1
            end
        i ←i+1
          path[I]← temp[p]
        b ← b-1
      End
   di ← destace(f)
End-algorithm
```

الجانب العملي

لقد تم بناء حقيبة برمجية تعتمد الخوارزميتين اعلاه لغرض اجراء التقييم للمسار الأفضل لاي مخطط ، حيث يكون المدخل هو شكل المخطط والناتج مصفوفة عناصرها المسار من عقدة البداية(initial state) وحتى عقدة الهدف(goal state). وفر النظام واجهة مستفيد تتكون من العناصر التالية:

١- Graph creation.

٢- Graph displaying.

٣- Find path using (Dijksra) shortest path.

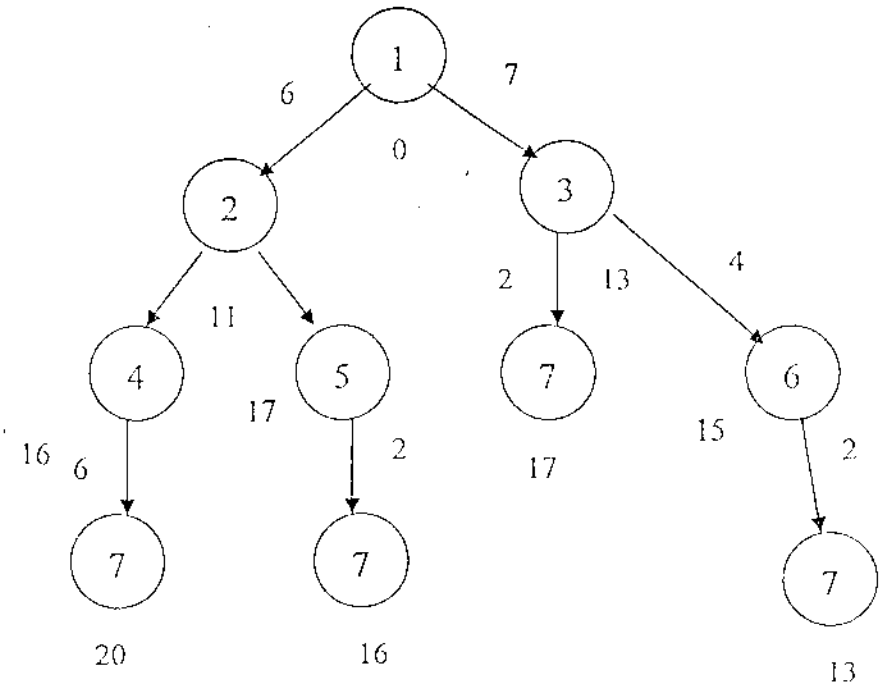
٤- Find path using (A^*) best path.)

٥- Exit

- يمثل الخيار الأول: عملية إدخال المخطط الذي سيتم البحث فيه و يكون الإدخال على شكل مصفوفة متجاورة (adjacency matrix) لخزن معالم المخطط .
- الخيار الثاني يمثل طريقة لعرض المخطط الذي قمنا بإدخاله على شكل جدول مفهرس لكل مدخل من مداخله يتم خزن طول الحافة أو قيمة المسار (cost) بين عقدتين في المخطط التي قد تكون مسافة أو زمن وغيرها وكذلك يخزن قيمة العقد (value) وهي قيمة معرفة .
- الخيار الثالث يمثل طريقة استخدام خوارزمية (Dijksra) لايجاد المسار مع ملاحظة ان هذه الخوارزمية تهتم بقصر المسافة بغض النظر عن قيمة العائق عند كل عقدة.
- الخيار الرابع يمثل طريقة استخدام خوارزمية (A^*) لأيجاد افضل مسار وهنا تم الأخذ بنظر الأعتبار وزن العائق لكل عقدة مزارة.

مثال:

لو كان لدينا المخطط التالي :



حيث تكون طريقة ادخال المخطط على شكل مصفوفة مؤشرات (Array of Pointers) لجميع العقد وتكون صيغة الأدخال على شكل سؤال وجواب كما يلي :

Q :Numbers of total nodes in the graph:

A: 10

Q :Enter Fist node and its value:

A: 1,0

Q: Enter Second node and its value:

A: 2,11

Q :Enter path cost between these nodes:

A :6

ويستمر الأدخال لكافة العقد في المخطط والقيم الخاصة بذلك. بعدها يتم السؤال على العقدة الهدف:

Q: Enter Goal node:

A: 7

ويستخدم الخيار الثاني في النظام لعرض المخطط اعلاه على الشاشة وكما يلي:

	1	2	3	4	5	6	7
1	0 10	6 11	7 13	∞ 16	∞ 17	∞ 15	∞ 17
2	∞ 10	0 11	∞ 13	2 16	4 17	∞ 15	∞ 17
3	∞ 10	∞ 11	0 13	∞ 16	∞ 17	4 15	∞ 17
4	∞ 10	∞ 11	∞ 13	0 16	∞ 17	∞ 15	∞ 17
5	∞ 10	∞ 11	∞ 13	∞ 16	0 17	∞ 15	∞ 17
6	∞ 10	∞ 11	∞ 13	∞ 16	∞ 17	0 15	∞ 17
7	∞ 10	∞ 11	∞ 13	∞ 16	∞ 17	∞ 15	∞ 17

ان هذه المصفوفة الموزونة تمثل المخطط اعلاه وهي بالأبعاد (7 x7) ومداخلها احد القيم الثلاثة التالية:

١- تكون قيمة المسار تساوي صفر(0) اذا كان الأنتقال من العقدة الى نفسها مع بقاء وزن العقدة فيها قيمة معرفة.

٢- اذا لم يكن هنالك مسار بين العقدتين فان قيمة المسار تمثل بالرمز(∞). ويتم اعطاء رقم ذات قيمة كبيرة مثلا (99999) في البرنامج مع بقاء وزن العقدة ثابت.

٣- في حالة وجود مسار بين العقدتين فان قيمة المسار هي وزنه مع بقاء وزن العقدة ثابت. وكما مبين في الجدول اعلاه .

عند تطبيق الفقرة الثالثة في النظام نلاحظ ان المسار الأقصر من العقدة (1) الى العقدة (7) يكون ذات وزن 9 ويحتوي العقد (7,3,1). حيث عند التنفيذ نعطي العقدة البدائية (initial state) وهي العقدة (1) والعقدة النهائية (final state) وهي العقدة (7) .

من الملاحظ ان هذه الخوارزمية تبحث في جميع عقد المخطط واوجدت اقصر مسار فعلي بين العقدة البدائية وعقدة الهدف ولكن لم تراعي وزن العائق عند كل عقدة ,وينحصر عمل الخوارزمية على قيمة وزن العقدة فقط تمثل قيمة المسار (cost) ومؤشرة في الحقل الثاني في الجدول.

اما عند تطبيق الخيار الرابع من النظام أي تشغيل (A*) فنلاحظ ان المسار الأمثل من العقدة البدائية (1) ولغاية عقدة الهدف (7) يكون بالوزن 24 وان عقد المسار هي (7,6,3,1).

ان هذه الخوارزمية تعمل بقيمة الحقلين في الجدول اعلاه قيمة المسار(cost) ووزن العقدة (value) اذ عائق .

من الملاحظ ان هذه الخوارزمية اعطت توازنا بين قصر المسار ووزن اقل عائق وهو الأمر المهم لتخطيط مسار الأنسان الالي. كذلك تم زيارة العقد التي تحتاجها في تخطيط المسار فقط.

ن الكلفة النهائية :

Total cost=24

S1=11

ناتجة من :

و S2=13 ناتجة من كلفة (cost =0) ووزن عائق (value =13).

المناقشة والأستنتاجات:

١- ان خوارزمية المسار الأفضل (Best path algorithm) قد لاتعطي المسار الأقصر دائما مقارنة بخوارزمية المسار الأقصر (Shortest path algorithm) ولكنها تهتم بقيمة العائق عند كل عقدة حيث تضع موازنة بين قصر المسار واقل قيمة للعائق والتي تمثل لنا طريق جيدة لتخطيط مسار الأنسان الالي والذي يمثل موضع اهتمام الباحثين.

٢- لا تكتفي خوارزمية المسار الأقصر بحل واحد لذا تقوم باختبار جميع العقد في المخطط بينما لاتستخدم الطريقة الأولى الا العقد التى تحتاجها لبناء المسار وتكتفي بحل امثل واحد فقط (Optimal solution).

٣- لم يصمم النظام البرمجي لبيئة مسارات ثابتة وانما يسمح بتحديث المخرجات اعتمادا على ماورد المخططات المدخلة الى النظام .

٤- وقت تنفيذ البرنامج يتناسب طرديا مع زيادة عدد العقد عندما يكون هنالك مخطط واحد للبيئة ككل بالأضافة الى التعقيدات الحاصلة اثنا عملية التحديث , اما عند افتراض المخطط كونه عدد من المخططات الجزئية فان وقت المعالجة لايتأثر بعدد العقد ولكن بعدد المداخل.

المصادر

1- DOORs, Quality systems and software Ltd, UK,1998, www.qssinc.com.
2-Elain Rich, Artificial Intelligence, international Edition , 1983.
3- I. McDonald, ISO TC184/SC4 N911,Application reference model for the Exchange of System Engineering Data, 2000.
4-Kamal G. U. P. & Angel P. Del. Pobil, Praerical Motion Planning in Robotics, 1998.
5-Michael , T. Goodrich & Roberto Tamassia, Data Structure and Algorithms in Java, 1998.
6- Parunak, H. V. D., Application s of distributed artificial intelligence in idustry' in: G. M. P. O'Hare, N.R.jennings (Eds.), 1996.
7- Patrick Henry Winston, Artificial Intelligence, 1981,1984.