人工知能･音声処理実験 作業日報

第2回 AI-2

学生番号:09430509

氏名:今田　将也

実施日:2020年11月05日

AI-2の概要

問題 3（優先発展♫）

- MATLAB/Octaveを用いた，Dijkstra’s algorithm の実装をした
- 迷路探索の結果として，1つ以上の最適経路探索の，探索結果を示した.
- 探索結果に対して，考察をした

問題 4（発展★）

- ゴールまでの経路出力を可能とした“深さ優先探索”と“幅優先探索”を実装し，考察を述べた

問題 3（優先発展♫）

最適経路探索として Dijkstra’s algorithm の実装をおこない，考察を述べよ.

問題3の小問題

1. MATLAB/Octaveを用いた，Dijkstra’s algorithm の実装を示しなさい.
2. 迷路探索の結果として，1つ以上の最適経路探索の，探索結果を示しなさい.（注：自作プログラムによって”結果”を得た，という事実がわかるような説明は必須とする.）
3. 探索結果に対して，考察せよ.（注：少なくとも「探索結果は想定通りか？」という観点での考察は必須とする.）
4. 様々なグラフを，Dijkstra’s algorithm で探索し，さらなる考察を加える

3-1. MATLAB/Octaveを用いた，Dijkstra's algorithm の実装を示しなさい(1/3)

- まずは，迷路問題を解くためのグラフに，コストの概念を導入した
  - 各要素の数値をコストとする
  - コストが小さいノードは経由しやすい
  - コストが大きいノードは経由しにくくなる
  - コスト0のみ「コスト0で接続されている」ではなく，「コスト0は，接続されていない」という意味
- コストに対応できるようにopen_listに入れるノードの構造体を拡張

```
open_list = [ struct('id', initial_node, 'parent', 0, 'cost', 0) ];
```

- 親ノードcur_nodeから，コストを引き継ぐ処理を追加

```
hyp_node_costs = A(cur_node.id, hyp_node_ids) + cur_node.cost;    % <-- new! [2a]
hyp_node_parents = repmat(cur_node.id, size(hyp_node_ids));
hyp_nodes = struct('id', num2cell(hyp_node_ids), ...
                   'parent', num2cell(hyp_node_parents), ...
                   'cost', num2cell(hyp_node_costs));          % <-- update! [2b]
```

3-1. MATLAB/Octaveを用いた，Dijkstra's algorithm の実装を示しなさい(2/3)

- オープンリストは，U-Priority Queue として実装
  - U-Priority Queueは追加するときにソート順を維持する

```
open_list = [open_list hyp_nodes];
[~, idx] = sort([open_list.cost]);              % <-- new! 値，何番目かが帰ってくる
open_list = open_list(idx);                     % <-- new!
[~, idx] = unique([open_list.id], 'first');     % <-- new!
open_list = open_list(sort(idx));               % <-- new!
```

- オープンリストでの重複のチェックは不要になったので，該当するソースコードをコメントアウト

- オープンリストに追加する際，hyp_node（探索候補の配列）空だと挿入する必要がないため，hyp_nodesを更新する前に事前検証処理を追加

```
% Validate: hyp_node_ids  <-- new
if isempty(hyp_node_ids)
  disp('No hypothesis: skip updating open_list')
  continue
end

PrepareNodes
  hyp_node_costs = A(cur_node.id, hyp_node_ids) + cur_node.cost;    % <-- new! [2a]
```

3-1. MATLAB/Octaveを用いた，Dijkstra's algorithm の実装を示しなさい(3/3)

- 終了条件も構造体のid要素で判別

```
if cur_node.id == target_node    % <-- update!!
    disp('I got Target node!')
    break
end
```

- cur_nodeが，構造体になったため，curnode.idに置き換え

```
hyp_node_ids = find(A(cur_node.id, :) > 0);  % <-- update!!
```

```
if ~isempty(open_list)
    hyp_node_ids = hyp_node_ids(~ismember(hyp_node_ids, [open_list.id]))  % <-- update!!
end
```

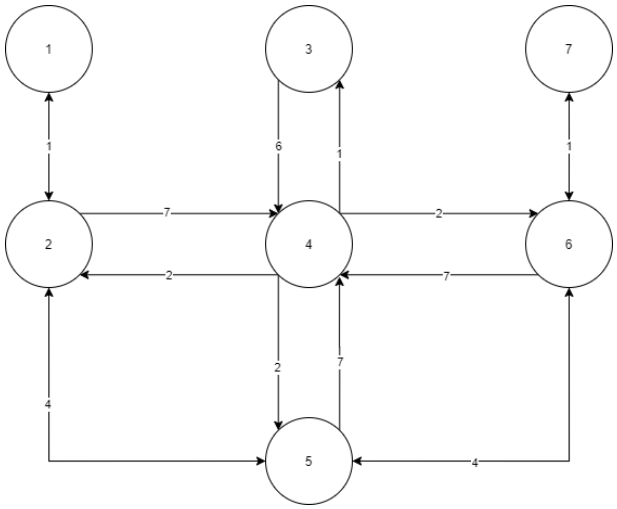
- open_list close_listが空のときの処理を記載

```
% closed_list実装により，展開済みのノードを再びオープンリストに格納しないようにする
if ~isempty(closed_list)
  hyp_node_ids = hyp_node_ids(~ismember(hyp_node_ids, [closed_list.id]));
end

% DeleteNodesInList: remove nodes in open_list
%    **ToDo:** This process will be omitted after 'AI-2'
%if ~isempty(open_list)
%    hyp_nodes = hyp_nodes(~ismember(hyp_nodes, [open_list.id]))
%end
```

3-2.迷路探索の結果として，1つ以上の最適経路探索の，探索結果を示しなさい．(1/2)

探索対象として，講義資料にあった有向グラフを用いた．以下その図である．



実験

- ✓全部で，7このノードからなる図である．
- ✓入るとき，出る時のそれぞれに重みを持たせた有向グラフになっている．
- ✓初期ノードは1，目標ノードは7で設定をした

```
A = [0 1 0 0 0 0 0;
     1 0 0 7 4 0 0;
     0 0 0 6 0 0 0;
     0 2 1 0 2 2 0;
     0 4 0 7 0 4 0;
     0 0 0 7 4 0 1;
     0 0 0 0 0 1 0;
];
initial_node = 1;
target_node  = 7;
```

3-2.迷路探索の結果として，1つ以上の最適経路探索の，探索結果を示しなさい．(2/2)

結果は以下のようになった．

```
DEBUG_closed_list =

    1     2     5     4     6     3     7
    0     1     2     2     5     4     6
    0     1     5     8     9     9    10

I got Target node!
route =

   1   2   5   6   7
```

【実験結果】

探索した結果として，

1-2-4-6-7のノードを通る結果が表示されていることがわかり，最適な探索が行えているように見える．

3-3.探索結果に対して，考察せよ.

【想定】

- 前述のグラフでは，重みの小さいところを通っていくのが最適な探索経路と言える.
- 目視では，1-2-5-6-7が最適な経路のように見える

【結果】

- 前述の通り，1-2-4-6-7のノードを通る結果が表示されていることがわかった.
- 想定通りの探索が行われていた.

問題3のまとめ

最適経路探索として Dijkstra’s algorithm の実装をおこない，考察を述べよ.

まとめ

- MATLAB/Octaveを用いた，Dijkstra’s algorithm の実装をし，その概要を理解した
- 迷路探索の結果として，1つ以上の最適経路探索の，探索結果を示し，その動作内容を見た
- 探索結果に対して，考察をし，想定とどうだったかについて述べた.

問題4

ゴールまでの経路出力を可能とした“深さ優先探索”と“幅優先探索”を実装し，考察を述べよ.

問題4の小問題

- 深さ優先探索や幅優先探索に対してどのように実装をしたか
- 幅優先探索あるいは深さ優先探索（または，その両方）による経路探索をおこない，その結果と考察を示す．

4-1深さ優先探索や幅優先探索に対してどのように実装をしたか

- 元はダイクストラ法による構造体を用いたものを使用
- オープンリストをソートする必要がないためその記述箇所を消去
- 展開済みのノードを再びオープンリストに格納しないようにするための処理を復帰

4-2.幅優先探索

【結果】

１－２－４－６－7を通る探索の結果になった.

```
DEBUG_closed_list =

    1     2     4     5     3     6     7
    0     1     2     2     4     4     6
    0     1     8     5     9    10    11

I got Target node!
route =

   1   2   4   6   7
```

【考察】

幅優先では最適な経路での探索はできない.

コストを検討しない探索方法であるからと考える

4-2.深さ優先探索

【結果】

１－２－４－６－7を通る，幅優先探索の結果と同様になった．

```
DEBUG_closed_list =

    1     2     4     5     3     6     7
    0     1     2     2     4     4     6
    0     1     8     5     9    10    11

I got Target node!
route =

   1   2   4   6   7
```

【考察】

深さ優先探索も幅優先探索と同じ結果であったため，最適な経路での探索はできないといえる

問題4のまとめ

ゴールまでの経路出力を可能とした“深さ優先探索”と“幅優先探索”を実装し，考察を述べよ.

まとめ

- 深さ優先探索や幅優先探索に対してどのように実装をしたかを述べた.
- 幅優先探索あるいは深さ優先探索（または，その両方）による経路探索をおこない，その結果と考察を示し，幅優先探索と深さ優先探索では，ダイクストラ法と同様の結果は得られないことがわかった.

AI-2の感想

理論だけ聞いていたダイクストラ法のアルゴリズムを実際のプログラムに落とし込んで，どのような実装が必要なのかについて実験を行えた．

Close_list，Open_listが空だったときの記載を忘れていて，同じノードが複数個入ってしまっていたが，再度きちんとアルゴリズムを見直すことでその間違いに気がつくことができた．

講義資料がわかりやすかったため，そのような間違いにも気がつくことができ，実験をある程度しっかりと行えたと思う．

あとはAI-1との内容をあわせて，ミニレポートを早急に完成させておきたい．