Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

Multi-goal Pathfinding in Cyber-Physical-Social Environments: Multi-layer Search over a Semantic Knowledge Graph

Oudom Kem*, Flavien Balbo, Antoine Zimmermann, Pierre Nagellen

Univ Lyon, MINES Saint-Étienne, CNRS, Laboratoire Hubert Curien, UMR 5516, F-42023, Saint-Étienne, France

Abstract

Multi-goal pathfinding (MGPF) is a problem of searching for a path between a start and a destination that allows a set of goals to be satisfied along the path. In this paper, we address MGPF in the context of ubiquitous environments such as airports, commercial centers and smart campuses that accommodate cyber, physical and social entities from smart objects, to sensors and to humans. The availability of data and services in such environments presents new opportunities for addressing MGPF. More precisely, given a MGPF problem in a pervasive environment, our approach exploits data from various resources, for instance, information acquired from cyber-physical entities located in the environment and from external resources such as the Web in order to solve the problem. In this approach, we propose a knowledge model for describing spatial structures of an environment, cyber-physical-social entities it contains, and relationships among them as a graph, called cyber-spatial graph (CSG). Depending on the set of goals to be satisfied, we dynamically build a graph of goal-location pairs (each goal paired with locations at which it can be satisfied), on top of a CSG creating a multi-layer graph. We adapt A* algorithm into a multi-layer A* that is able to search on both layers. We propose heuristics that exploit the structure and knowledge of CSG to improve the search. Our experiments show significant time efficiency and node expansion reduction in various graph structures when employing the heuristics.

Peer-review under responsibility of KES International.

Keywords: Multi-goal Pathfinding; Search algorithms; Semantic Web; Cyber-Physical-Social environment

1. Introduction

In this information age, from smart phones, to intelligent artificial personal assistants and to smart homes, we are experiencing the shifting towards ubiquitous computing, Internet of Things (IoT) and artificial intelligence. Cyber-physical entities are embedded in social environments of various scales from smart homes, to airports, to smart cities, and the list continues. This paradigm shift supplies us with tremendous amount of useful information and services, thus presenting opportunities to address classic problems in new, different, and potentially more efficient manners.

Multi-goal pathfinding (MGPF) is a problem of searching for a path between a start and a destination that allows a set of goals to be satisfied along the path. In this paper, we address MGPF in the context of ubiquitous environments accommodating cyber, physical and social (CPS) entities such as sensors, smart objects and humans. The aim of our

* Corresponding author. Tel.: +33-477-426-619
E-mail address: oudom.kem@emse.fr

Peer-review under responsibility of KES International.

approach is to solve MGPF by exploiting data acquired from CPS entities in a given environment and from external resources such as the Web. To understand the underlying motivation of our approach, consider the following scenario. A traveler, Bob, arrives at an airport. Bob wants to find a path to his departure gate. Bob has a set of activities (goals) he wants to do on his way to the gate: get a trolley for his luggage, check-in, buy a takeout for lunch and find a waiting seat near a power plug to charge his laptop. Using spatial information of the airport, we can find a path to the gate. Information about the airport makes it possible to determine which locations allow Bob to satisfy each of his goals. For example, restaurant is a business which prepares and serves food and drinks to customers in exchange for money. By obtaining that piece of information from the Web, we are able to deduce that Bob can buy lunch at locations of type restaurant. Dynamic and up-to-date information from sensors and smart objects enables us to determine the optimal path for Bob. For instance, instead of going to a trolley area, which is at the opposite direction of his gate, it is possible to locate an available trolley nearby that was left by other people, thanks to data from connected trolleys. We might suggest Bob to take an escalator instead of an elevator because we know that there are too many people in the queue waiting for the elevators or that the elevators are out of service thanks to the feeds from sensors. In addition, information from social entities such as other travelers or personnel can be used to enhance Bob's travel experience. For instance, reviews by travelers (e.g. quality or availability) on restaurants enable us to choose locations that are at Bob's best interests and preferences. Other interesting goals can be to find a person in an airport or to look for help in case of language difficulty, just to name a few.

Considering the aim of the approach, one might ask two challenging questions: (1) Which resources to use to solve a MGPF problem? (2) How to deal with the dynamics, mobility and heterogeneity of CPS entities? The first question is concerned with the discovery of resources that are relevant to a MGPF problem. We address this question via the use of a data model to capture necessary knowledge enabling resource discovery. Regarding the second question, there are existing works that address these issues in the context of IoT. As an example, in[1], the authors propose a multi-agent-based socio-technical network (STN) to manage the complexity of CPS entities. In this paper, our focus is on the conceptual level, and we employ one of the existing solutions such as STN to abstract away the complexity at the lower level. Furthermore, the state of an environment is dynamic. Referring back to the previous example, at time t_0, the elevator might be available, but at t_1, it might become occupied. Such dynamics has influence on the path. Given the fact that an action of satisfying a goal (e.g. buy lunch, check-in) takes a certain amount of time, an optimal pre-planned path may lose its optimality over time. This necessitates en-route planning to keep refining the initial path according to up-to-date information. In this paper, our focus is on pre-trip pathfinding. Our contributions include a knowledge model for abstracting a ubiquitous environment integrating spatial and CPS dimensions, an ontology for describing the model in a machine-readable and extensible manner, an abstraction model integrating goals and spatial dimension as a multi-layer graph, knowledge-based heuristics for improving search processes and a multi-layer A* algorithm adapted for searching over a multi-layer graph.

The rest of this paper is structured as follows. First, we provide a background on multi-goal pathfinding and discuss related work. Second, we present an abstraction of a ubiquitous environment for MGPF, and formally define MGPF. Third, we present our approach to solving MGPF. Fourth, we present and discuss experimental results of the approach. Finally, we conclude the paper, and discuss the limitations of the current approach along with some promising directions to address these limitations.

2. Background and related work

In literature, there are two common variants of MGPF. First, given a single start and multiple goals, MGPF is defined as a problem of searching for paths for each start-goal pair, resulting in multiple paths[2]. Second, MGPF is treated as a traveling salesman problem (TSP) in which the aim is to find a path from a start to a number of goals before reaching a destination such as in[3,4]. The problem addressed in this paper is to some extent similar to the second definition. However, unlike the classical TSP, we have constraints on the order of goals to satisfy. The work in[5] addresses a TSP with order constraints, called partially ordered TSP. The author proposes two algorithms to solve the selection and ordering of points-of-interests, which are places, for indoor navigation systems. The computation of a path is based on complete spatial knowledge of an environment, and distance is the sole criterion for path evaluation. In this paper, we address the partially ordered TSP, but the specific property of our problem is that satisfying a goal is not limited to passing by a place, but can be any activity carried out via a cyber, physical and/or social entity, which

can be mobile and dynamic, in the environment. Our approach uses up-to-date and dynamic information retrieved from various resources for path computation. We use generic criteria, not limited to distance, and quality of entities, determined using qualitative information from resources, for path evaluation.

3. Problem domain

In this work, we assume that the topology of a real-world ubiquitous environment, denoted by E, is represented as a grid map of atomic locations, called tiles. A search graph of an environment can be built from the grid map, defining each tile as a node. A directed edge between two adjacent nodes n and n' is defined if, in the given environment, the location represented by n' is directly accessible from that by n. We use E to denote a ubiquitous environment, and define it as $E = \langle SG, CPSE, R \rangle$ where:

- SG is a search graph defined as $SG = \langle L, C \rangle$ where L is a finite set of locations in E, and $C = \{c^{i-j}\}$ are connections between locations with $i \in L$ being the origin location and $j \in L$ the destination location.
- $CPSE$ is a finite set of CPS entities located in E. A CPS entity is defined by $cpse = \langle l, t \rangle$ where $l \in L$ is the location at which $cpse$ is located at time t. $CPSE$ includes any cyber, physical and/or social entities in a given environment. For instance, in an airport, physical entities can be gates and restaurants; cyber-physical entities can be sensors and connected objects; social entities can be humans or agents, both software and hardware, equipped with social capability [1].
- $R = \{r^n\}$, where $n \in C \cup CPSE$, is a finite set of resources providing information about a $cpse$ or information on how to move between locations. An example of a resource can be a website, a database, a social network or an API to sources of data collected from various cyber-physical entities.

Spatial information is not sufficient to determine at which location a goal can be satisfied. Up-to-date and qualitative information is also needed to find an optimal path. These reasons motivate our approach to exploit data from various resources. This brings us to a challenging question: which resources to use to solve a MGPF problem? MGPF consists of 2 main aspects - the goals and the path. We can then establish two different kinds of relationships: (1) between $CPSE$ and R - resources providing information about a CPS entity, (2) between L and R - resources providing information on how to move between locations. However, knowledge about these kinds of relationships is non-existent. To capture such knowledge, we propose a knowledge model, entitled cyber-spatial model, that integrates spatial and CPS dimensions of an environment with their relationships to resources.

3.1. Knowledge model

In this model, the representation of an environment is organized as a hierarchy, denoted by HE, that can be composed of more than one levels. Each level consists of a set of hierarchy entities. A hierarchy entity $he \in HE$ is a spatial division in an environment. Locations are grouped into different hierarchy entities. The model describes an environment as a multigraph, called cyber-spatial graph and denoted by CSG. Fig. 1 shows an example of such a graph. The top layer is the spatial dimension of an environment represented as a search graph. The bottom layer consists of resources R relevant to an environment. The middle layer, description of an environment as a CSG, integrates spatial and CPS dimensions of an environment together, and connect them to relevant resources. It is essential to note that there could be multiple edges between two location nodes, and each of these edges c is labeled with a resource $r^c \in R$.

Our objective is to enable software agents to make use of such representations by providing them with machine-readable descriptions of the environments upon which they may reason to discover more knowledge. These reasons motivate us to use semantic web principles and technologies as the foundation of our knowledge model. More precisely, the model is based on RDF[6] (Resource Description Framework). RDF facilitates the integration of heterogeneous data with different underlying schema, and is compatible for representing distributed graphs; hence, along with the use of IRI (Internationalized Resource Identifier), CSG of an environment can be stored in different fashions from centralized manner to widely distributed on the Web. In both case, CSG enables humans and agents to exploit its knowledge and facilitates the integration with CSGs of other environments. To provide concepts and properties

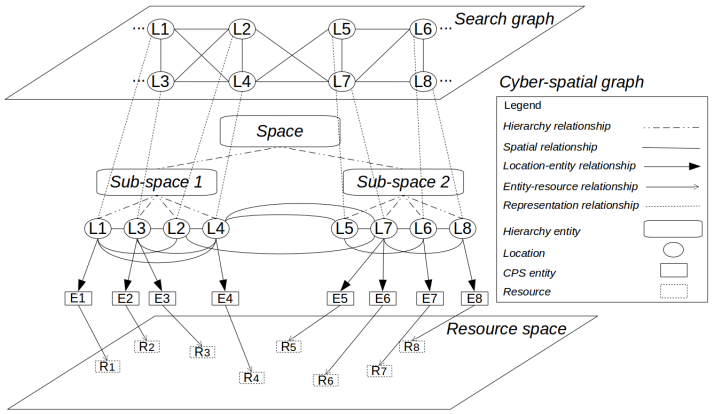

Fig. 1. An example of a cyber-spatial graph

necessary to describe the knowledge in our model, we design an ontology, entitled cyber-spatial ontology[1], using OWL[7] (Web Ontology Language). An overview of the ontology is given in Fig. 2. Throughout this paper, we use `cso` as a prefix binded to the ontology's IRI. The main concepts defined in this ontology are CPS entity, location and hierarchy entity.

The class `cso:CPSEntity` is defined as the class of CPS entities (*CPSE*) in an environment. The `cso:isOfType` property represents a unidirectional connection that relates a CPS entity to a CPS entity type, represented by the class `cso:CPSEntityType`. For instance, a CPS entity type can be sensor, mobile object or business place. Respectively, a CPS entity can be an elevator sensor, a connected trolley or a burger shop. The `cso:hasRelevantResource` property relates a CPS entity to resources that provide information about that entity. For example, a resource relevant to a restaurant can be that restaurant's website. A resource of an elevator sensor can be an API to a sensor data store.

The class `cso:Location` represents locations (L) in an environment. The `cso:containsEntity` property describes the relationship between a location and a CPS entity located in that location at a given point in time. Locations are grouped into hierarchy entities. Each location is directly under a hierarchy entity (`cso:HierarchyEntity`). This type of relationship is described by the `cso:isUnder` property. As an example, a location $l \in L$ contains ten sensors, two gates and five trolleys (`cso:CPSEntity`) at time t. l is directly under the coverage of terminal 1. A location can be directly connected to other locations. The class `cso:Connection` represents a direct connection from a location to another. Each connection is attributed with a finite set of resources providing information about the path between the two locations. The property `cso:hasConnectingResource` relates a connection to such a resource. For instance, a direct connection from l to $l' \in L$ is attributed with a set of resources $R^{l-l'} \subset R$ providing information about the path from l to l'. An example of a location described using this ontology is shown in Fig. 3.

The class `cso:HierarchyEntity` represents all hierarchy entities (HE) in an environment. The `cso:hasLevel` property relates a hierarchy entity to a level of hierarchy, represented by the class `cso:HierarchyLevel`. For example, in a hierarchy that consists of three hierarchy levels - airport, terminal, zone, a hierarchy entity can be a terminal, a zone or an airport. A hierarchy entity may directly contain a finite set of locations. The relationship between a hierarchy entity and a location directly under that entity is described by the `cso:containsLocation` property. A hierarchy entity may be further divided into a finite set of hierarchy entities of a lower level. An example can be a terminal 1 consisting of zone 1, 2 and 3. To express this kind of connections, we use the property `cso:containsHierarchy`. Furthermore, a hierarchy may also be directly under another hierarchy of a higher level (parent hierarchy). Taking the preceding example, terminal 1 can be under an airport. To relate a hierarchy entity to its parent hierarchy, the property `cso:belongsTo` can be used. Such relationship is non-existent if the hierarchy entity is of the highest level (i.e. root). A hierarchy entity also contains a set of CPS entity types located under its coverage, which can be described by using the property `cso:containsEntityType`. The rationale behind this is to facilitate the subgraph extraction algorithm presented later in section 4.1. A hierarchy entity $he \in HE$ may be connected to other hierarchy entities of the same

[1] https://partage.mines-telecom.fr/index.php/s/5KeuwFuvFjD4RjS

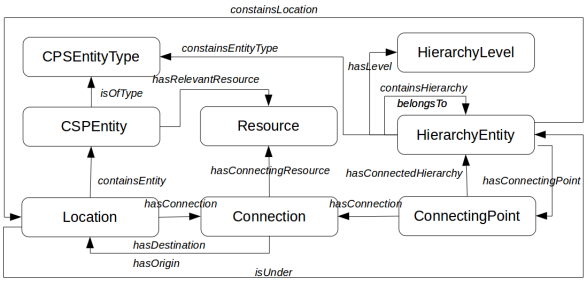

Fig. 2. An overview of the cyber-spatial ontology

```
@prefix cso:  <http://simulated.org/ontologies/2017/1/cso#> .
@prefix sim:  <http://simulated.org/location/> .

sim:Location_1
        cso:isUnder             <http://simulated.org/hierarchy/Terminal_1> ;

        cso:containsEntity      <http://simulated.org/cps-entity/Restaurant_A> ;
        cso:containsEntity      <http://simulated.org/cps-entity/Gate_10> ;
        cso:containsEntity      <http://simulated.org/cps-entity/Trolley_201112> ;
        cso:containsEntity      <http://simulated.org/cps-entity/Roof_sensor_12192> ;

        cso:hasConnection       [ cso:hasOrigin                 sim:Location_1 ;
                                  cso:hasDestination            sim:Location_2 ;
                                  cso:hasConnectingResource     <http://c-resource-i.com/> ] ;
        cso:hasConnection       [ cso:hasOrigin                 sim:Location_1 ;
                                  cso:hasDestination            sim:Location_10 ;
                                  cso:hasConnectingResource     <http://c-resource-k.com/> ] .
```

Fig. 3. An example of a location description

level. To describe how he is directly connected to $he' \in HE$ where he and he' are at the same level, we use the class `cso:ConnectingPoint`. A connecting point is defined by a connected hierarchy entity he' and a connection c, instance of the class `cso:Connection` where:

- the origin of c is an exit point l^{exit} of he where $l^{exit} \in L$ is under the coverage of he
- the destination of c is an entry point $l^{entry} \in L$ of he' where l^{entry} is under the coverage of he'
- a finite set of resources $R^{he-he'} \subset R$ for the path from l^{exit} to l^{entry}

A ubiquitous environment is dynamic, so its CSG needs to be updated to take into account the dynamics. The current state of CSG is used as an input for solving MGPF problems in the environment.

3.2. Problem formulation

We formally define multi-goal pathfinding $MGPF$ as $MGPF = \langle CSG, n_o, n_d, G, C \rangle$ where:

- CSG is a cyber-spatial graph, as previously defined
- $n_o \in L$ is a node representing a start location
- $n_d \in L$ is a node representing a destination
- G is an ordered list of goals to satisfy. A goal $g \in G$ is an activity to be satisfied, and g may be satisfied at a number of locations $L^g \subset L$.
- C is a set of problem-specific criteria for evaluating a path. For instance, a criterion can be distance or price.

A problem is solved when an optimal path is found. A path is a list of locations through which every goal can be satisfied in the given order. A path is optimal if it has a minimum cost evaluated based on C.

4. The approach

An overview of our approach to MGPF is shown in Fig. 4. The order in which goals G are satisfied can be determined in two different fashions. First, the order is given explicitly as a part of the problem. An example of such case is when a user explicitly specifies the order of goals in which they want to accomplish. Second, the order is not explicitly specified, but needs to be determined due to the constraints of the goals. In this approach, we assume the first case in which input goals are already in the right order.

The first step of the approach is to associate each goal to the locations at which it can be satisfied (Algo 1 - Line 2). A goal $g \in G$ can be satisfied at a location $l \in L$ if l contains one or more CPS entities that enable g to be achieved. This is where the semantics comes into effect. In the knowlege model, previously described in section 3.1, each instance of `cso:CPSEntity` is associated with a *CPS entity type*, an instance of the class `cso:CPSEntityType`. To determine whether a goal can be satisfied at/using a CPS entity, the key idea is to find additional data on the Web about the type of the CPS entity in a follow-your-nose fashion[8]. Though essential, this step is not in the scope of this paper. We focus on the next three steps - subgraph extraction, goal-space graph generation, and multi-layer search.

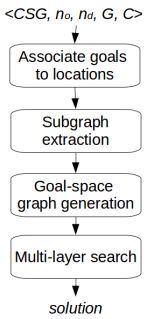

Fig. 4. An overview of the approach

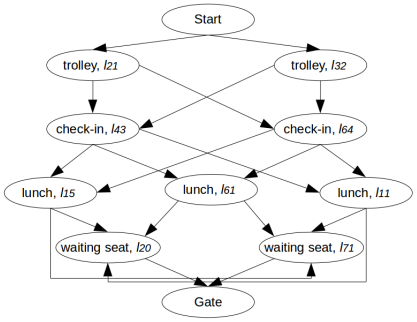

Fig. 5. An example of a goal-space graph

4.1. Subgraph extraction

The objectives of this process are, first to identify locations in *CSG* where one or more goals can be satisfied, and second, to extract a subgraph from *CSG*, in which all the goals in G can be satisfied. The rationale behind subgraph extraction is to reduce the size of the graph over which we execute search algorithms to that of the one containing only relevant nodes. In addition, the extracted knowledge will also be used in heuristics of the search algorithm, which will be discussed later in the paper. Pseudo code of the extraction algorithm is shown in Algorithm 1 and 2. Algorithm 1 takes *CSG* and G as inputs. First, it identifies the CPS entity types associated with each goal (Algo 1 - Line 2). Second, it traverses the tree to extract relevant hierarchy entities. This is done by using the function *search* (Algo 2) that takes as inputs the IRI of a hierarchy entity and a list of CPS entity types associated to the goals. This function returns a list of child nodes (locations or hierarchy entities) that contain CPS entities of types specified in the list. The output of the algorithms are (1) locations where each goal can be satisfied (goal-location pairs) and (2) a subgraph containing locations allowing all goals to be satisfied.

4.2. Goal-space graph generation

Although the order of goals to be satisfied is known, a goal may be satisfied at multiple locations. To determine which location is optimal for satisfying each goal, we need to take into account the following factors: the cost for moving from the current goal's location to the next goal's location and the quality of CPS entities in the next goal's location. The quality of a CPS entity is evaluated using qualitative information from resources. Referring back to Bob's scenario, reviews from other customers on a restaurant can be considered as qualitative information for evaluating restaurants according to Bob's constraints. To represent the space of this problem, we construct a goal-space graph by using G and the goal-location pairs extracted from the preceding step.

A goal-space graph, denoted by π, is an acyclic graph where nodes are goal-location pairs, and they are connected according to the order of goals defined in G. This graph allows us to represent goal-location relationships in the required order. It can be viewed as a partial plan in which there may be multiple goal-location pairs for each goal. We use the definition of a partial plan in [9], and define $\pi = (GV, GE, L)$ where:

- GV is a set of nodes of an acyclic graph. Each node $v \in GV$ contains a goal $g \in G$ and a location $l \in L$ where g can be satisfied at l.
- GE is the edges of an acyclic graph. They represent ordering constraints on the goals in GV. We define $v \prec v'$ if $v \neq v'$ and (GV, GE) contains a path from v to v'.
- L, as previously defined, is a set of locations in a given environment E

For illustration purpose, refer back to the previous scenario about Bob. Bob's goals consist of trolley, check-in, lunch and waiting seat near a plug. Suppose the followings are locations where Bob's goals can be satisfied: trolley = $\{l_{21}, l_{32}\}$, check-in = $\{l_{43}, l_{64}\}$, lunch = $\{l_{15}, l_{61}, l_{11}\}$ and waiting seat = $\{l_{20}, l_{71}\}$. We can generate a partial plan π as shown in Fig. 5. π is an abstract graph built on top of *CSG* to integrate the goal dimension, creating a multi-layer

graph. An edge of π is equivalent to a path that may consist of multiple nodes on *CSG*. For instance, an edge between $(lunch, l_{15})$ and $(waitingseat, l_{20})$ may be equivalent to a sequence of nodes $(l_{15}, l_{16}, l_{28}, l_{18}, l_{20})$ on *CSG*.

At this stage, we have the necessary data including a *CSG*, a subgraph that covers locations relevant to *G*, and a goal-space graph or partial plan π. The remaining step is to determine an optimal path by using the knowledge.

Algorithm 1 extract(CSGHierarchyRoot, goals)

```
for all goal in goals do
    entityType ← goalToCPSEntityTypeMapping(goal)
    add (entityType, goal) to entityTypeGoalMap
previousAreas ← [ ]
currentAreas ← [CSGHierarchyRoot]
currentIRIs ← [CSGHierarchyRootIRI]
while currentAreas not empty do
    previousAreas ← currentAreas
    currentAreas ← [ ]
    childrenIRIs ← [ ]
    for all iri in currentIRIs do
        areas ← search(iri, entityTypeGoalMap)
        if areas not empty then
            for all area z in areas do
                add z to currentAreas
                add iri of z to childrenIRIs
    currentIRIs ← childrenIRIs
goalToPlaces ← [ ]
for all area z in previousAreas do
    currentGoals ← get goals of z from entityTypeGoalMap
    for all goal g in currentGoals do
        insert (g, z) in goalToPlaces
return goalToPlaces
```

Algorithm 2 search(iri, entityTypeGoalMap)

```
candidateAreas ← [ ]
for all cpsEntityTypes t present at the iri do
    if t is in entityTypeGoalMap then
        potentialChildren ← get from iri the list of children containing t
        for all child c in potentialChildren do
            add c to candidateAreas
return candidateAreas
```

Algorithm 3 $A^*_\pi(\pi, n_{start}, n_{dest}, CSG)$

```
openList ← [n_start]; closedList ← []; cachedPathCost ← []
while openList not empty do
    n ← pop-min(openList)
    if n is the destination n_dest then
        return n
    else
        generate n's successors and set their parents to n
        for all n' successor of n do
            g ← cachedPathCost(l_n → l_n')
            if g is null then
                path_n-n' ← A*_CSG(CSG, l_n, l_n', cachedPathCost)
                g ← path_n-n'.cost
            g(n') ← g + n.g
            h(n') ← h_π(n')
            n'.f = g(n') + h(n')
            if n' not in openList nor closedList then
                add n' to openList
            else if n' in openList and f(n') < f(n'_openlist) then
                replace n'_openlist by n'
            else if n' in closedList and f(n') < f(n'_closedlist) then
                remove n'_closedlist from closedList
                add n' to openList
```

Algorithm 4 $A^*_{CSG}(CSG, l_{start}, l_{dest}, cachedPathCost)$

```
openList ← [l_start]; closedList ← []
while openList not empty do
    l ← pop-min(openList)
    if l is the destination l_dest then
        return l
    else
        generate l's successors and set their parents to l
        for all l' successor of l do
            g ← request-resource r^l-l'
            add [(l → l'), g] to cachedPathCost
            h ← h_goal(l') + h_hierarchy(l')
            l'.f = l.g + g + h
            if l' not in openList nor closedList then
                add l' to openList
            else if l' in openList and f(l') < f(l'_openlist) then
                replace l'_openlist by l'
            else if l' in closedList and f(l') < f(l'_closedlist) then
                remove l'_closedlist from closedList
                add l' to openList
```

4.3. Multi-layer search

A goal-space graph is an abstract graph that is constructed based on a *CSG*. Computing the cost of an edge between two nodes of a goal-space graph necessitates a pathfinding between two corresponding nodes on a *CSG*. Therefore,

the search space is composed of two different graphs - a goal-space graph π and its underlying *CSG*. We consider each graph as a layer, making it a multi-layer graph.

4.3.1. Heuristics

Heuristics play an important role in graph search. Good heuristics lead a search process to find a path quicker and expands less nodes. In our context, node expansion can be costly. Expanding a node requires the algorithm to compute the cost function of every reachable neighbor. Considering it needing access to resources to retrieve information for the computation, the number of expands has a significant impact on the overall performance of the algorithm. For example, accessing resources might introduce latency or problems with threshold limits in accessing resources' API. Therefore, we propose heuristics that exploit the knowledge of each layer to improve the search process.

On the π layer, we design a heuristic h_π for evaluating each node that is computed based on the number of goals that can be satisfied at the node's location and the quality of CPS entities the location contains. To evaluate a CPS entity, we retrieve information from resources relevant to the entity, as described using the property `cso:hasRelevantResource`. The rationale behind using such information is that there may be multiple locations where each goal can be satisfied. Introducing qualitative information as a part of the cost function allows us to find the best location for each goal.

On the *CSG* layer, we propose two different heuristics, namely h_{goal} and $h_{hierarchy}$. Goal-based heuristic h_{goal} uses the subgraph extracted during subgraph extraction stage. The heuristic prioritizes nodes that are in the subgraph. The logic behind this heuristic is that the subgraph contains locations and hierarchy entities in which goals can be satisfied. Expanding those nodes may lead to more promising paths as they are the locations or under the hierarchy entities where at least one goal can be satisfied. Let CSG_{sub} be an extracted subgraph, L_{sub} a finite set of locations in CSG_{sub} and HE_{sub} a finite set of hierarchy entities in CSG_{sub}. If a node n belongs to L_{sub} or n's direct or indirect hierarchy entity he_n belongs to HE_{sub}, we prioritize n by assigning its heuristic cost $h_{goal}(n) = 0$; Otherwise, $h_{goal}(n) = 1$. Hierarchy-based heuristic $h_{hierarchy}$ prioritizes expansion of nodes that are located in a hierarchy entity closer to that of the destination. The underlying motivation of this heuristic is that moving between two nodes in the same hierarchy is less costly than between nodes in different hierarchies, with cost being generic (e.g. distance, time, price). In practice, let n_{dest} be the destination node and n be the node being evaluated. At each level of the hierarchy, if n's hierarchy entity is different from that of n_{dest}, we add a 1 to n's heuristic cost $h_{hierarchy}(n)$. For example, referring back to Fig. 1 and assuming that $L7$ is the destination node, $h_{hierarchy}(L5) = 0$ because $L7$ and $L5$ share the same hierarchy entity at every level. $h_{hierarchy}(L1) = 1$ because at the second level, $L1$ is under a hierarchy entity *Sub-space 1*, while $L7$ is under *Sub-space 2*.

4.3.2. Multi-layer search algorithm

Many algorithms have been proposed to address pathfinding problems such as Dijkstra's algorithm, uniform-cost search and the most well-known A* algorithm. The aim of A* is to find a minimum-cost path. It evaluates nodes by their cost using a cost function $f(n) = g(n) + h(n)$ where $g(n)$ is the cost to reach n from the start node and $h(n)$ is the minimal estimated cost to reach a destination node from n. This property of A* allows us to use the knowledge-based heuristics we propose for each layer of a multi-layer graph. To search over a multi-layer graph, we adapt A* algorithm into a multi-layer A* (MLA*) to handle resource accesses and allow communication between π layer and *CSG* layer.

In MLA*, we combine two different versions of A*, an A^*_π for searching on the π layer and an A^*_{CSG} for the *CSG* layer. A^*_π (Algo 3) behaves similarly to classic A* except during node expansion. When a node n is expanded, a set of nodes reachable from n is found. However, as the nodes are abstract, n's location l_n and each of its neighbors' locations $l_{n'}$ might not be directly connected. To compute the path between n and each of its neighbors, A^*_π instantiates an A^*_{CSG} to search on *CSG* for a path between l_n and $l_{n'}$. To evaluate each node, A^*_π uses a cost function $f_\pi = g_\pi + h_\pi$. The cost of movement g_π on π layer is obtained from executing an A^*_{CSG}.

On a *CSG* layer, A^*_{CSG} (Algo 4) is also based on the classic A* with a modification to communicate cached results of pathfinding (i.e. path and costs) to A^*_π so that those results might be reused. It is necessary to note that on the *CSG* layer, the search space is composed of locations, while hierarchy entities are used for computing the heuristics. To evaluate a node, A^*_{CSG} uses a cost function $f_{CSG} = g_{CSG} + h_{CSG}$. To compute g_{CSG}, we use information from connecting resources of each location, described using the property `cso:hasConnectingResource`. As for the heuristic cost h_{CSG}, we combine two heuristics, namely h_{goal} and $h_{hierarchy}$.

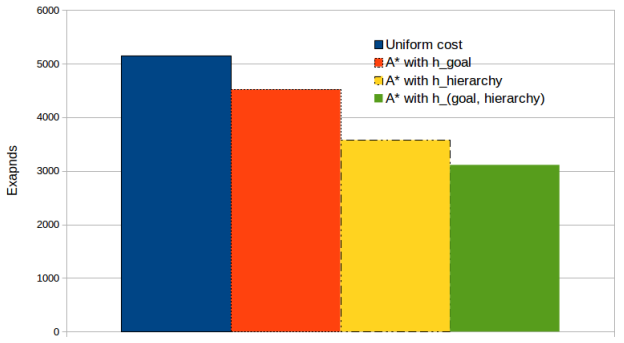

Fig. 6. Expands of A* using different heuristics on the same graph

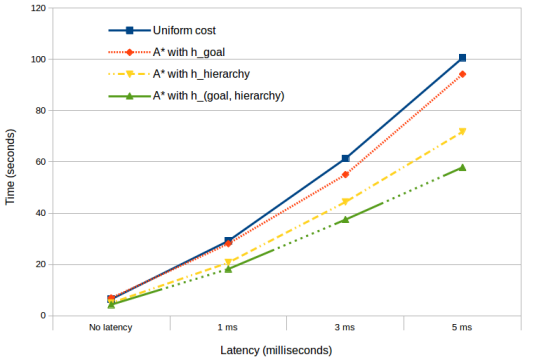

Fig. 7. Run time of A* using different heuristics on the same graph

5. Experimental results

Our experiments were executed on a 2.4GHz Intel Core i7 laptop with 16GB of RAM. We use three different *CSGs* for these experiments. Graph 1 has a 3-level depth hierarchy (1 hierarchy entity at first level, 10 at second, 100 at third), 10000 nodes (locations) and 10000 CPS entities. Graph 2 and 3 have a 4-level depth hierarchy (1 hierarchy entity at first level, 10 at second, 100 at third, 1000 at fourth), 10000 locations and 10000 CPS entities. The difference between graph 2 and 3 is that in graph 2, there is only one connection between 2 locations and 1 exit point for each hierarchy entity, while there are 3 connections and 2 exit points in graph 3. For each experiment, we used 2 types of requests: the start and destination nodes are (1) in the same hierarchy entity, (2) in different hierarchy entities.

First, we conducted an experiment with graph 3 using 3 different heuristics, namely h_{goal}, $h_{hierarchy}$, and both of them combined h. We use uniform-cost algorithm as a baseline algorithm for comparison. h expands the least reducing almost 40% of expanded nodes compared to the uniform-cost algorithm, followed by $h_{hierarchy}$ and then h_{goal} as shown in Fig.6. We introduced latency in accessing resources, which are 1, 3 and 5 milliseconds (ms), during node expansion. For instance, in the 3 ms case, we generate latency values between 0 to 3 ms. The time efficiency increases in function of the latency, depicted in Fig. 7.

In the second experiment, we focused on different types of requests over the same graph (graph 1). Fig. 8 demonstrates time efficiency (%) gained by using A* with h compared to uniform-cost. The results show that for requests of type (1), uniform-cost is more efficient than A* even with latency. This is because when a request involves only one hierarchy entity, $h_{hierarchy}$ has no effects. In addition, A* is slower because it spends time computing the heuristics. For type (2) requests which involve multiple hierarchy entities, A* starts to overtake uniform-cost at around 2 ms of latency.

In the third experiment, we compared A* with h to uniform-cost algorithm over 3 different graph structures, graph 1, 2 and 3, as illustrated in Fig. 9. There is a significant difference in efficiency over different graph structures. We believe that the heuristic is more efficient in graph 3 because there are more exits and more connections between nodes. In such case, the heuristic guides the search to choose better nodes among the options. In graph 2 and 3 however, there are not many options. From each node, only one path is possible. Comparing graph 1 and graph 2, the heuristic works better on graph 2 because locations are more distributed. In graph 1, each 1000 locations are grouped under the same hierarchy, so the $h_{hierachy}$ has a minimal effect. Uniform-cost outperforms A* when there is no latency for graph 1 and graph 2 because the number of expands reduced cannot compensate for the time A* needs to compute its heuristic functions. However, when there is latency, the time needed to expand each node increases, making A* gain more time efficiency compared to uniform-cost.

Concerning the efficiency, the uniform-cost algorithm performs better in most cases when there is no latency. The reason is the uniform-cost algorithm does not use any heuristic, while A* spends more time to compute its heuristic function. However, A* starts to surpass uniform-cost when latency, around 1 ms, is introduced. This time efficiency is linear to the introduced latency as shown in Fig. 9.

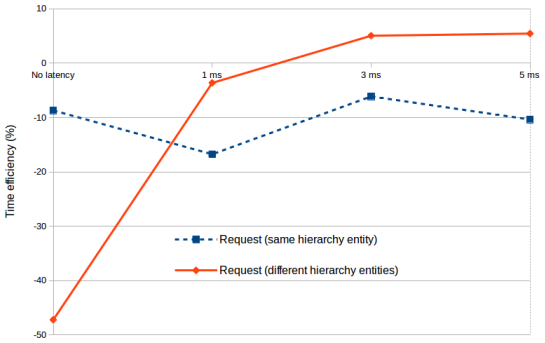

Fig. 8. Run time efficiency of A* with $h_(goal, hierarchy)$ on graph 1

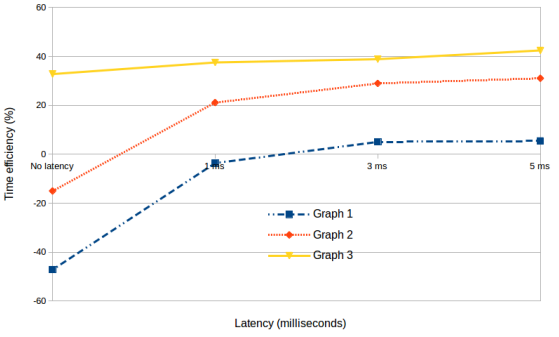

Fig. 9. Run time efficiency of A* with $h_(goal, hierarchy)$ compared to uniform-cost over different graph structures

In terms of effectiveness of the approach, in all our experiments with various graph structures, requests, types and numbers of CPS entities, we obtained the desired solutions with minimum cost. Though not a definite proof of optimality, A* with our heuristics did provide the optimal solution as uniform-cost in all cases of our experiments.

6. Conclusion

This paper describes an approach to multi-goal pathfinding in a ubiquitous environment. The approach uses data from cyber-physical entities in an environment as well as from external resources such as the Web to solve multi-goal pathfinding problems. We discussed the missing knowledge about connections between spatial and CPS dimensions of environments. We proposed a knowledge model for semantically describing such knowledge. We presented a method to generate a goal-space graph based on a given set of goals and a *CSG*. A goal-space graph is an abstract graph built on top of a *CSG*, making it a multi-layer graph. We proposed different heuristics for the algorithm, each of which exploits the knowledge described in *CSG*. An adaptation of A* algorithm was made to address the search over multi-layer graphs. It is worth mentioning that our knowledge model is not limited to describing a single environment. One of our objectives is to allow representations of different environments to be compatibly integrated together, thus rendering the representations extensible and making our approach applicable in solving MGPF problems of different scales. In our future work, we will address the latency issue resulted from accessing resources. Our first intuition is to parallelize the algorithm to concurrently search. In addition, information in *CSG* is dynamic. For instance, cyber-physical things such as smart trolleys are mobile. Dynamically updating *CSG* is important for the accuracy of solutions. Therefore, en-route pathfinding will be addressed as a part of our future work.

References

1. Ciortea, A., Zimmermann, A., Boissier, O., Florea, A.M.. Towards a social and ubiquitous web: A model for socio-technical networks. In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*; vol. 1. 2015, p. 461–468.
2. Lim, K.L., Yeong, L.S., Ch'ng, S.I., Seng, K.P., Ang, L.M.. Uninformed multigoal pathfinding on grid maps. In: *2014 International Conference on Information Science, Electronics and Electrical Engineering*; vol. 3. 2014, p. 1552–1556. doi:10.1109/InfoSEEE.2014.6946181.
3. Berbeglia, G., Cordeau, J.F., Laporte, G.. Dynamic pickup and delivery problems. *European Journal of Operational Research* 2010;**202**(1):8 – 15.
4. Codognet, P.. *Multi-Goal Path-Finding for Autonomous Agents in Virtual Worlds*. Boston, MA: Springer US; 2003, p. 23–30.
5. Werner, M.. Selection and ordering of points-of-interest in large-scale indoor navigation systems. In: *2011 IEEE 35th Annual Computer Software and Applications Conference*. 2011, p. 504–509. doi:10.1109/COMPSAC.2011.71.
6. Cyganiak, R., Wood, D., Lanthaler, M.. RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014. 2014. URL: `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/`.
7. Hitzler, P., Krtzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.. OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation 11 December 2012. 2012. URL: `https://www.w3.org/TR/owl2-primer/`.
8. Heath, T., Bizer, C.. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology* 2011;**1**(1):1–136.
9. Ghallab, M., Nau, D., Traverso, P.. *Automated Planning and Acting*. Cambridge University Press; 2016.