Journal of Computational Science xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Application of distributed graph transformations to automated generation of control patterns for intelligent lighting systems

Igor Wojnicki*, Leszek Kotulski, Adam Sędziwy, Sebastian Ernst

AGH University of Science and Technology, Department of Applied Computer Science, Al. Mickiewicza 30, 30-059 Krakow, Poland

ARTICLE INFO

Article history:
Received 22 March 2017
Received in revised form 14 July 2017
Accepted 16 September 2017
Available online xxx

Keywords:
Graph transformations
Rule-based systems
Outdoor lighting
Optimized design
Intelligent control

ABSTRACT

Designing a large infrastructure, such as a street lighting system, is a complex task itself especially in the context of Smart City and Smart Grid approaches. The problem is made even harder if it needs to be designed with control in mind. To facilitate a complex design process without losing fidelity, a graph-based formalism, namely General Environment Model (GEM), is proposed to be applied to model such an environment. Moreover, another graph-based model, namely Control Availability Graph or shortly CAG, is proposed to enable definition of routines for dynamic control of large-scale systems. Both of these models have been verified in practice, but the transition from GEM to CAG had been performed manually. In this paper, we propose a coherent, formal method of generating a control system from the graph-based environment description while taking into account the designer's decisions. An application of the generated CAG as a control system yields up to 34% of energy consumption reduction in a pilot deployment of over 3500 light points for the city of Krakow, Poland.

1. Introduction

There are numerous ways to describe architectural or technological spaces. Such spaces are made up of all the components which are part of their surroundings, including buildings, streets, pavements, tracks, bridges, power grid, street lighting, transportation etc. These components are described by certain parameters. The parameters could be either properties of a particular component or its location. Formalizing such a description makes further analysis possible [13]. Such an analysis can lead to parameter optimization, which in turn results in operation or deployment cost reduction. It is especially applicable regarding modern large-scale solutions, such as Smart City and Smart Grid initiatives [32], where not only scale but also complex interactions among environment components are important. It also delivers a base for establishing intelligent control if needed.

This paper focuses on formal representation of the space and methods used to process it, with architectural and outdoor lighting components as an example. As a result, a coherent formal graph-based representation is proposed. It is subsequently used to establish intelligent control of the outdoor lighting system. As an example a gas station lighting is chosen. Similar case was investigated previously in [3] but in the context of determining an optimal lighting solution (fixture type and power) rather than developing a control system.

Spatial relationships among 3D environment elements are described using a graph representation called General Environment Model (GEM) [37,38] denoted by $G \in L(GEM)$, where G is an element of the language generated by the graph grammar $\mathcal{GEM}$. In other words, a scene represented by G may be obtained as a result of graph grammar derivation (see Definition 2). This type of graph can be automatically generated from GIS (Geographic Information System) data [27].

The second field of application of graphs and their transformations is control of a lighting system. In [36], the knowledge necessary for an AI control system to make the correct decision is also represented by a graph called Control Availability Graph (CAG) denoted by $H \in L(\mathcal{CAG})$. It defines functional relationships among elements of a lighting system, where $\mathcal{CAG}$ is a proper graph grammar that allows to control the lamps based on information from sensors. As for now, the graph H has to be defined individually and "manually" for a scene defined by G. Our goal is to efficiently transform[1] a scene graph G into a control graph H or, more precisely, to generate H on the basis of G. The demand of efficiency is

* Corresponding author.
E-mail addresses: wojnicki@agh.edu.pl (I. Wojnicki), kotulski@agh.edu.pl (L. Kotulski), sedziwy@agh.edu.pl (A. Sędziwy), ernst@agh.edu.pl (S. Ernst).

[1] In general, graph isomorphism is a NP-hard problem.

http://dx.doi.org/10.1016/j.jocs.2017.09.011
1877-7503/© 2017 Elsevier B.V. All rights reserved.

important due to the size of the graph, which represents the scale of the problem – in practical examples, the graph can contain thousands of nodes, and its manual conversion would not be feasible. Furthermore, scenes under consideration may change in time, so a way to automatically update H is needed. This transformation has been verified in the real urban environment consisting of more than 3500 light points.

2. Motivation

The growing share of solid state lighting and the properties of LED lamps, most of all the extremely short onset time and customizable photometric solid, motivate engineers to search for energy savings. However, these properties are not fully exploited yet. In particular it refers to the rudimentary control capabilities of lighting installations, offered by commercially available systems such as: CityTouch by Philips, Owlet by Schreder or LightGrid by GE. A main application of such systems is to provide the time schedule based dimming and access to technical properties of the luminaires regarding their condition and energy usage. It has to be kept in mind that one deals here with a large system covering thousands of lighting points. Thus even small unit optimization leads to tangible savings at the city scale. Let us consider a city with 25,000 streetlights and a unit power optimization of 2 W per lamp. Suppose, for the rough estimation, that the installation is switched on for 8 h a day. The annual energy savings in such a case is 146 MW h.

The key difficulty with applying a dynamic lighting control is computational overhead related to processing sensor input data (ambient light level, actual traffic intensity, pedestrian presence and others) and transforming it into control requests being sent to particular streetlights. It is present both at the design stage, while developing control schemas and programming control devices, and during actual control operations.

To resolve the computational overhead issue related to the scale of a lighting installation one has to use a computer to make a design process fully assisted and automated. Any intermediate action performed by a human, while creating a lighting design, becomes a process' bottleneck. Prior to using any algorithmic, computer aided, method, a domain of a problem needs to be formalized. To do this we propose using hypergraph models covering entire environment which includes both its spatial properties and control actions performed on it. What is more, using such a model enables parallelization of computing which provides scalability. The latter advantage is very important for time-effective computing and thus for practical application of the proposed method. In previous works we showed that graph and hypergraph representations are suitable for enclosing both a structure of an environment and its static and dynamic properties such as traffic flow, presence of objects and so on (see [25,26]). As shown in [28] thanks to graph-based parallel computing accompanied by auxiliary heuristics one can reduce computing time by a factor of 26 for preparing a photometric design. It was also shown, that applying an adaptive, dynamic control, based on traffic intensity data instead of predefined schedules, yields 34% of additional energy savings [35]. In this case it was only a single street for which the CAG was prepared manually. For larger installations, e.g. city scale, such an effort would be infeasible. That is why we want to show that a graph structure describing physical entities and relations among them – GEM – can be successfully used to define an adaptive control for large-scale lighting installations. It is achieved by transforming GEM into CAG.

3. Related research

Contemporary intelligent lighting research focuses on indoor lighting. It includes both office and home spaces. Sensors deployed in predefined areas are used to obtain information about the state of the environment. Control actions are performed taking into consideration user lighting preferences, inhabitant tracking, daylight intensity detection and window blinding control which leads to energy savings [20,21]. Data from actual experiments in an office space is also presented in [16,1]. Outdoor lighting optimization for tunnels is discussed in [10]. It is based on control schemas using data obtained from vehicle presence and luminance detectors. Additionally some luminance requirements are given to comply with safety regulations. Other areas that can lead to energy consumption reduction, such as forestation of portal surroundings [23], or delivering sunlight using optical fiber [24] are also researched.

Some experiments with outdoor lighting has also been carried out. Taking into consideration both traffic and weather conditions up to 40.9% energy savings have been obtained on a highway [12].

All approaches presented above indicate a common theme: possible energy savings due to intelligent control, which is often confirmed by the experiments. However, it needs to be pointed out that they regard small-scale implementations only. They do not tackle scalability issues which is necessary for a wider, e.g. city-scale deployment. In particular each light point lighting level and its adjustments have to be photometrically calculated according to the norms (see [15,6,4]). Such a compliance with the norms makes our environment safe at night, which is required.

4. 3D environment representation

To enable efficient analysis and modeling of large-scale or complex systems, the description of the system needs to be formalized. In the considered context, we will use graph models to express system properties and behavior. There are several graph representations of 3D spaces available, ranging from the simplest ones, where particular objects are modeled by vertices and their spatial relations are given by edges, up to highly expressive hierarchical hypergraph models which may describe complex built-up areas. The hypergraph model, which is the primary representation in further considerations, is broadly accepted and used in 3D modeling [7,2], so its further discussion is beyond the scope of this paper. Moreover, the problem of selection of a particular graph model for a 3D scene is secondary from our perspective.

Besides hypergraphs, we will use the “naive” graph model which is obtained as a result of hypergraph aggregation. A basis of the hypergraph approach, details of which may be found in [29], and its transition to the regular graphs is sketched below.

4.1. Hypergraphs

Let T will be a polyhedron consisting of n faces $(f_1, f_2, \ldots, f_n)$, k edges $(a_1, a_2, \ldots, a_k)$ and m vertices $(h_1, h_2, \ldots, h_m)$. Hypergraph H_T representing the polyhedron T is a tuple $H_T = (V, A, H)$, where $V = \{f_1, f_2, \ldots, f_n\}$ is a set of hypergraph nodes (physical faces), $H = \{h_1, h_2, \ldots, h_m\}$ is a set of hypergraph hyperedges (physical vertices), and $HA = \{a_1, a_2, \ldots, a_k\}$ is a set of hypergraph edges (physical edges). Fig. 1 presents the hypergraph model of a cuboid. In case of two adhering polyhedrons, we introduce an additional, logical type of edge referred to as an *external edge* which denotes adherence of two faces.

In some circumstances, we require general information about an overall object structure, rather than its detailed properties. In such cases, it is sufficient to have the knowledge about relations among particular components of a scene. To achieve that, we transform (*collapse*) hypergraphs to special vertices using *aggregating transformations*. Such an operation, also referred to as *synthesis*, is described in [30]. All data describing how external edges were attached to the collapsed hypergraph are stored as values of supple-