Coherent Configuration and Operation of Building Transportation Systems

Peter B. Luh*, Laurent Michel**, Eugene Santos Jr.**, Danqing Yu*, Andrew See**,
Bo Xiong*, Greg Johnson**, Shi Chung Chang*

*Electrical and Computer Engineering, **Computer Science and Engineering
University of Connecticut
Storrs, Connecticut 06269-2157 or 2155**, USA
Peter.Luh@uconn.edu, ldm@engr.uconn.edu, Eugene@engr.uconn.edu

Abstract – Configuration and operation of building transportation systems, e.g., elevators and stairs for offices, hotels, and apartments, are important, and have profound societal impact such as in improving efficiency, reducing costs, and saving lives. Establishing methodologies that are effective and coherent across configuration and operation phases, however, is difficult. In this paper, coherent configuration and operation of building transportation systems is studied through a synergistic integration of optimization, formal semantics, and constraint satisfaction. Building on a formal semantics, a statistical method for elevator configuration using a coarse-grain model and a newly developed optimization method for elevator operation using a fine-grain model are developed. In view of their different underlying assumptions, different model granularities, and the methods themselves, there is a lack of coherency between these two methods. A new approach is then established that synergistically integrates the two methods based on constraint programming. Testing results demonstrate that the new approach efficiently selects high quality configurations with performance coherent across model granularities and configuration/operation phases.

Index Terms – Building transportation systems, Coherent configuration and operation, Group elevator scheduling.

1. INTRODUCTION

Configuration and operation of building transportation systems, e.g., elevators and stairs for offices, hotels, and apartments, are important, and have profound societal impact such as in improving efficiency, reducing costs, and saving lives. The problems, however, are difficult. A configuration, once implemented, cannot be easily changed, and may have life-long impacts on operations. Effective configuration must therefore consider various aspects of operations with sufficiently realistic models with multiple scopes and resolutions and a web of interdependencies. The gap between configuration and operation, however, hinders systematic translation and consistency, causing specifications satisfied in the configuration phase to come up short during the operation phase. The effective integration of methods and solutions to form a coherent set of solutions has proven to be difficult.

In this paper, coherent configuration and operation of building transportation systems under normal conditions is studied. Given a building layout and its intended use, the problem in the configuration phase is to determine the specifications of elevators (including the key decision variables of speed, capacity, and the number of elevators) and stairs to optimize a given performance measure subject to budgetary and other relevant constraints. For a given configuration, the problem in the operation phase is to determine the dispatching of elevators and the use of stairs to optimize a certain performance measure. Our goal is to establish an effective and computationally efficient methodology that is coherent across configuration and operation phases as illustrated in Figure 1.
differences are caused by their different underlying assumptions, different model granularities, and the methods themselves. In particular, the statistical method may not be accurate when utilization is high, or when multiple decision variables are changed in different directions. In view of the above, a new approach is then established that synergistically integrates the two methods to select high quality configurations through constraint programming (Van Hentenryck, 1987, Van Hentenryck, Deville, and Teng, 1992, Michel and Van Hentenryck, 1997 and 2002). Testing results demonstrate that the new approach efficiently selects high quality configurations with performance coherent across configuration and operation phases.

The overall flow of our solution concept is illustrated in Figure 2 where formal semantics (FS) and coherence enforcement (CE) by using constraint programming are highlighted.

![Figure 2. Schematic for the Quality and Coherent Solution Framework](image)

2. LITERATURE REVIEW

System Configuration

Product configuration has been investigated by those manufacturers who moved from mass production to producing semi-customized or customized products. Manufacturers of computers, automobiles, or telecommunication equipment, to name a few, now depend on their ability to customize products based on customer-specified requirements using standardized components. Approaches used include rule-based reasoning, model-based reasoning, and most notably case-based reasoning and constrained-based configuration (see Sabin and Weigel, 1998 for a good survey). Current research focuses on capturing user preferences (Junker, 2003), incremental configuration, and reconfiguration. For building transportation systems, the major challenge is the configuration of elevators. The standard practice is to decompose a building into zones, and design elevators for each zone by using formulas derived empirically or from simplified analytical models, e.g., queueing models (Barney and dos Santos 1977, Strakosch 1998, Howkins 1998, Powell 2002, and Barney 2003). There is no method that guarantees the satisfaction of operational performance for a given configuration. More effective and coherent methods are needed.

System Operation

The dispatching of a group of elevators has long been recognized as an important problem. However, it is difficult in view of hybrid system dynamics, combinatorial state and decision spaces, and time-varying and uncertain passenger demand. Recent results include reinforcement learning (Crites and Barto, 1996 and 1998); AI planning methods (Koehler and Schuster, 2000 and Koehler and Ottinger, 2002); and optimization-based approach by using queueing analysis with specific traffic patterns (Pepyne and Cassandras, 1997 and 1998) or dynamic programming with incremental optimization (Nikovski and Brand, 2003a, b). All these papers assume that the configuration under consideration is given, and do not address configuration issues.

From the above, it can be seen that building configuration and operation problems have mostly been treated separately, and many with limited scopes or performance. In view of the increasing complications of modern buildings, opportunities introduced by advanced information technology and mathematical optimization, and increasingly stringent performance requirements imposed by customers, a coherent treatment of both problems is becoming imperative.

3. SOLUTION METHODOLOGY

Formal Semantics

Formal semantics are provided by an ontology, which has been studied since the 1990’s and has received recent attention in computer science for application to the Semantic Web (Maedche 2002). The fundamental role of an ontology is to provide the vocabulary for describing a target domain, in this case, building transportation systems, including knowledge about buildings, building transportation systems, and relevant solvers. The ontology also explicitly represents implicit assumptions and “common sense” knowledge (e.g., inches and meters are both units of measurement for length). This semantic information is necessary to determine coherency across different levels of resolution as well as in the interaction of system components.

To be more precise, the ontology stores information about building transportation systems and their properties. For example, the ontology would provide the knowledge that an elevator has a capacity in people, maximum speed, type (passenger or freight), area consumed per floor and a cost. It also specifies a range for domain variables, for instance the capacity of an elevator in people is a positive integer. Many of the constraints used in constraint programming described later are derived from knowledge within the ontology. The ontology also supplies the knowledge to differentiate between aspects of different building systems at multiple levels of resolution.

Applying semantic knowledge to the interaction of system components ensures coherence in the interaction of
solvers. For example, the ontology can determine if a particular unit of measure is valid for a specified system property (e.g., ensure a variable specified in seconds represents time duration). Additionally, knowledge can be retrieved from the ontology to perform conversion between units of measure where necessary (e.g., converting car speed in feet per minute to meters per second). The ontology thus ensures that various modules and solvers can be integrated coherently. Furthermore, with knowledge of the input and output of a solver, the ontology helps determine the necessary variables and constraints for the constraint programming module. This lends flexibility in system architecture where solvers can be added with minimal integration effort.

**Statistical Methods**

Statistical methods have been widely used to evaluate the performance of group elevators. Under appropriate simplifying assumptions on passenger arrivals and elevator operations, these methods are easy to implement, computationally efficient, and can provide generalized conclusions. For example, a traditional procedure is to determine an adequate number and capacity of elevators for a given traffic pattern and elevator rated speed to satisfy certain performance criteria (e.g., average passenger wait time should be less than or equal to 20 seconds) (Barney, 2003).

In this paper, a coarse-grain statistical model is established to estimate the average passenger wait time (AWT) for different elevator configurations as depicted in Figure 3.

![Figure 3. The Schematic of Performance Estimation](image)

<table>
<thead>
<tr>
<th>Building Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of floors</td>
</tr>
<tr>
<td>Average inter-floor height</td>
</tr>
<tr>
<td>Arrival rate (persons/sec.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Elevator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of elevators</td>
</tr>
<tr>
<td>Elevator capacity</td>
</tr>
<tr>
<td>Speed and acceleration</td>
</tr>
<tr>
<td>Door open and close times</td>
</tr>
<tr>
<td>Load and unload times</td>
</tr>
<tr>
<td>Cost and space usage</td>
</tr>
</tbody>
</table>

The method is based on Barney (2003) with the following assumptions:

- The building configuration is given, where inter-floor distances are identical.
- Each elevator configuration contains multiple elevators of a particular type, where each type is characterized by a rated speed and capacity.
- The traffic pattern is up-peak where all passengers enter the elevator system from the lobby. The reason is that the ability to handle the up-peak traffic pattern generally implies the ability to handle other traffic patterns such as down-peak and inter-floor.
- Passengers arrive at a constant rate.
- Destination floors are uniformly distributed.
- Elevators arrive at the lobby at constant intervals for pick-ups. It is assumed that elevator capacity is sufficient to accommodate new arrivals during the interval between successive elevator arrivals.
- Other time delays (e.g., passenger disturbance) are negligible.

The key steps to derive the average wait time for a particular configuration is as follows (based on Barney 2003):

1. Calculate the round trip time for a single elevator.
2. Calculate the time interval (T) between successive elevator arrivals at the lobby.
3. Evaluate the average wait time based on T and the expected elevator load.

To calculate the round trip time for a single car, the expected reversal floor, the number of stops on the way up, and utilization (should be less than 95%) are first derived from building and traffic information. The round trip time is then calculated based on the inter-floor distance, car speed and acceleration, door open and close times, passenger load and unload times, and an initial estimate of the time interval between successive elevator arrivals at the lobby $T^0$.

For the second step, the time interval (T) between successive elevator arrivals is obtained as the round trip time divided by the number of elevators. This time interval T is generally different from the initial estimate $T^0$. A procedure is thus developed to update T by iterating through Steps 1 and 2 until T converges to $T^*$.

Using the converged time interval between successive elevator arrivals $T^*$, the average wait time (AWT) is then obtained based on queueing analysis and simulation (Barney and Santos, 1977). For example, when utilization (U) is in-between 60% and 80%, there is a linear relationship between AWT and $T^*$:

$$ AWT = [0.4 + (1.8U-0.77)^2] T^*.$$  (1)

Through simple analysis it can be established that the average wait time decreases monotonically as any of the three key decision variables (speed, capacity, and the number of elevators) increases.

Some of the assumptions presented above may not hold in practice. For example, passengers may not arrive at a constant rate, and elevator trips may have different durations which result in unequally spaced arrivals at the lobby. Furthermore, the operation of elevators depends heavily on the scheduling method. Fine-grain models with realistic scheduling methods are thus needed to reliably evaluate elevator configurations.

**Optimized Group Elevator Scheduling**

Our fine-grain model and the corresponding optimization method are based on the work of Xiong, Luh,
and Chang, 2005. In the formulation, key characteristics of group elevator scheduling are abstracted to establish an innovative two-level formulation, with passenger-to-car assignment at the high level, and the dispatching of individual cars at the low level. Passenger-to-car assignment constraints are established as linear inequality constraints, and are “coupling” constraints since individual cars are coupled by serving a common pool of passengers. Car capacity constraints and car dynamics are embedded within individual car simulation models. The objective function is flexible within a range of passenger-wise and car-wise measures, e.g., average wait time or service time over a planning horizon.

To solve the problem, the objective function is first transformed into an additive form, and the converted problem is decomposed into individual car subproblems through the relaxation of passenger-to-car assignment constraints. As shown in Figure 4, subproblems are independently solved by using a local search method in conjunction with dynamic programming to optimize single car dispatching with a novel definition of stages, states, decisions, and stage-wise costs. Within the surrogate optimization framework (Zhao, Luh, and Wang, 1999), a subproblem solution “better” than the previous one is “good enough” to set multiplier updating directions. Individual cars are then coordinated through the iterative updating of multipliers by using surrogate optimization for near-optimal solutions.

Constraint Programming

Constraint programming (CP) is a relatively new paradigm in the field of programming languages that seeks to separate the problem description from the solution strategy. Problems are specified as a system of constraints over finite domain variables, and the goal is to find an assignment of values to variables satisfying all constraints. In the solution strategy, the search component typically consists of an implicit enumeration and backtracking search while constraints are used for pruning variable domains to reduce the search space. In this work, Constraint Programming is used to model the configuration design problem and to establish coherence between models. It is used to search for a configuration that satisfies the specifications represented by constraints and rank these feasible configurations based on an objective function. A more concrete and formal Constraint Programming model is presented next.

For a given list of elevator types that includes the speed, acceleration, capacity, area, and cost of each elevator, the configuration problem reduces to deciding the elevator type and the number of elevators used. The decision variables are e.Type and NbElevators while e.Speed, e.Accel, e.Cap, e.Area, and e.Cost represent dependent variables. The domains of these dependent variables are given by the input list of possible elevator types and are linked by constraints to the decision variable e.Type. Additional variables are used to represent feasibility conditions. For instance, variable TotCost denotes the financial cost of the configuration and is constrained by:

\[ \text{TotCost} \leq \text{max\_cost} \]
\[ \text{TotCost} = \text{NbElevators} \times \text{e.Cost} \]

to meet the budget requirements. Similarly, variable TotArea denotes the floor area used and is constrained by:

\[ \text{TotArea} \leq \text{max\_area} \]
\[ \text{TotArea} = \text{NbElevators} \times \text{e.Area} \]

Configurations are further restricted to those that meet the service requirements with the constraint

\[ \text{AverageWaitTime} < \text{max\_wait\_time}. \]

Finally, the evaluateWaitTime constraint uses statistical and optimization estimators to evaluate configurations and enforces the relationship between average wait time and the elevator group configuration:

\[ \text{evaluateWaitTime}( \text{e}, \text{NbElevators, awt} ) \]

where e is the elevator object described above. The constraint uses the monotonicity property of the average wait time with respect to the number of elevators, the elevator speed, and capacity to efficiently prune the domains of variables. For instance, a projection on the speed allows us to find the smallest speed that yields a satisfactory average wait time, and to prune all configurations with a smaller speed. This, in turn, triggers the pruning of the e.Type variable. Similar projections on other variables lead to the pruning of e.Cap, and NbElevators.

![Figure 4. The Decomposition and Coordination Method](image-url)
One advantage of using Constraint Programming is the ease with which new constraints can be added and composed, without any change to the search procedure. Once relationships between different models or subsystems are derived, constraints can enforce those relationships allowing the larger problem to be solved without modifying the models of individual subproblems.

4. TESTING RESULTS

The goal of our testing is to establish coherence across the coarse-grain and fine-grain resolutions provided by the statistical and optimizing estimators as well across the configuration and operation phases of a building. The concrete problem is to select the top three configurations, in terms of performance, from a set of about 70 possible configurations, taking into account budget and space constraints. In view that the optimization estimator uses detailed operation models to generate its estimates, the ranking of its configurations is used as the standard to establish coherence across the configuration and operation phases.

Table 1 presents the top ten configurations from the statistical estimator and Table 2 shows the performance determined by the optimizing evaluator using statistical ranking as the indexing variable. For the optimization method, 50 Monte Carlo simulation runs are performed to obtain an estimate of AWT and the corresponding standard deviation. It can be seen that the AWTs for both estimators are monotonic in any of the three key decision variables of speed, capacity, and the number of elevators as presented in Section 3. However, when two decision variables are changed in different directions at the same time, the rankings of the two methods may not be consistent. For example, the configurations ranked second and third by optimization were ranked fifth and sixth by the statistical estimator and have one fewer elevator than those ranked first and second by the statistical estimator.

Table 2. Evaluation Comparison of Statistical and Optimization Estimator

<table>
<thead>
<tr>
<th>Rank</th>
<th>Statistical Estimator</th>
<th>Optimization Estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.19</td>
<td>7.19</td>
</tr>
<tr>
<td>2</td>
<td>9.02</td>
<td>9.90</td>
</tr>
<tr>
<td>3</td>
<td>9.54</td>
<td>10.23</td>
</tr>
<tr>
<td>4</td>
<td>9.84</td>
<td>10.66</td>
</tr>
<tr>
<td>5</td>
<td>10.3</td>
<td>8.21</td>
</tr>
<tr>
<td>6</td>
<td>10.33</td>
<td>9.67</td>
</tr>
<tr>
<td>7</td>
<td>11.2</td>
<td>14.30</td>
</tr>
<tr>
<td>8</td>
<td>13.26</td>
<td>14.36</td>
</tr>
<tr>
<td>9</td>
<td>13.47</td>
<td>10.06</td>
</tr>
<tr>
<td>10</td>
<td>14.44</td>
<td>15.22</td>
</tr>
</tbody>
</table>

In view of the above, our aim is to integrate these two methods within Constraint Programming to obtain results that are consistent with the true ranking by taking advantage of the much faster runtime of the statistical estimator. While using the `evaluateWaitTime` constraint with the statistical estimator, it was observed that the computational overhead of the Constraint Programming library does not cause a noticeable slowdown in generating feasible configurations. It was also observed that this constraint was able to significantly prune the domains of variables using the estimated AWT, thereby reducing the size of the search space. This suggests that the optimization estimator could be similarly used to prune the search space.

The `evaluateWaitTime` constraints with the statistical and optimization estimators could be integrated in two ways: (strategy AGREE) removing configurations only when the two constraints agreed that the configuration’s performance was not sufficient, and (strategy EITHER) removing configurations when either constraint determined that the configuration should be pruned, running the
statistical estimator first for efficiency. The motivation for the AGREE strategy is that it preserves some solutions that have a better true AWT than predicted by the statistical estimator. On the other hand, the EITHER strategy should provide additional pruning but at the risk of eliminating some good configurations.

Because of the fast speed of the statistical method as compared to the optimization method, a two-phase approach has been developed. First, the statistical method is used to rank all potential configurations, filtering out configurations whose performance are significantly worse then the allowed max_wait_time. This ranking is then used to guide the search for the top three configurations while using the evaluateWaitTime constraint with the optimization evaluator. The motivation is to get reliable estimates from optimization, while reducing the number of times optimization needs to be called. In particular, if a configuration differs in only one key decision variables of speed, capacity, and the number of elevators from one higher in the ranking we say that it is dominated by the higher ranked configuration. In this case the dominated configuration does not need to be evaluated by optimization because the property of monotonicity guarantees that the dominated configuration will not be any better. Strategies to further exploit domination along multiple dimensions are currently under investigation.

5. CONCLUSION

In this paper, coherent configuration and operation of building transportation systems is studied through a synergistic integration of optimization, formal semantics, and constraint satisfaction. It is demonstrated that there is a lack of coherency between the statistical method and the optimization method in view of their different underlying assumptions, different model granularities, and the methods themselves. A new approach is established that synergistically integrates the two methods based on constraint programming. Testing results demonstrate that the new approach efficiently selects high quality configurations with performance coherent across model granularities and configuration/operation phases.

The above results are generic and will have broad impact on diverse application domains where configuration and operation design and evaluation are a non-trivial task. Our effort will be the first to provide a systematic environment for complex evaluation and synthetic benchmarking, and will help designers better understand the impact of decisions and ultimately make the best ones.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation grant DMI-0423607.

REFERENCES