Computer Intensive vs. Heuristic Methods in Automated Design of Elevator Systems

Leopoldo Annunziata, Marco Menapace, Armando Tacchella

KEYWORDS

Computer-Intensive Modeling and Simulation, Artificial Intelligence, Computer Automated Design of Physical Systems.

ABSTRACT

Automated design of systems may require modeling and simulating potential solutions in order to search for feasible ones. This process often involves a trade-off between heuristics and computer-intensive approaches. Since neither of the two methods guarantees to always succeed, each problem domain requires a dedicated evaluation. In this paper, the domain of computer-automated design (CautoD) for elevator systems is studied with the goal of providing experimental evidence about which approach is best in which circumstances, and to serve as guidance for automated modeling of elevator systems.

INTRODUCTION

Automated design of physical system — see, e.g., [ZWPG03] — is the process whereby a project of some implement is carried out by computer programs which partially substitute the work of engineers and technicians. At the highest level of automation, the process requires a designer to enter configuration parameters, guidelines and physical constraints only, and the burden of generating feasible designs will rest on computer programs. Since physical systems involve the combination of several different elements to perform their stated function, the problem often becomes combinatorial in nature. In particular, when confronting a huge number of alternative designs, the question arises as to whether heuristics or computer-intensive methods should be leveraged. While heuristics tend to be less demanding in terms of computing power, they might also fail to explore potentially fruitful directions; computer-intensive methods, on the other hand, do explore more solutions than heuristics to achieve accurate results at the expense of computing power. It is well known that heuristics — see, e.g., [Pea84] — cannot guarantee desirable properties such as optimality with respect to some cost function, or completeness with respect to some set of solutions. However, computer-intensive methods often fail to deliver because of an excessive request of computational resources. Since neither of the two methods guarantees to always succeed, each problem domain requires a dedicated evaluation.

In this paper, computer-automated design (CautoD) of elevator systems is considered. Computer-automated design (CautoD) differs from "classical" computer-aided design (CAD) in that it is oriented to replace some of the designer's capabilities and not just to support a traditional work-flow with computer graphics and storage capabilities. While CautoD programs may integrate CAD functionalities, their purpose goes far beyond the replacement of traditional drawing instruments and most often involves the use of advanced techniques from artificial intelligence. As mentioned in [BOP+16], the first scientific report of CautoD techniques is the paper by Kamentsky and Liu [KL63], who created a computer program for designing character-recognition logic circuits satisfying given hardware constraints. In mechanical design — see, e.g., [RS12] — the term usually refers to tools and techniques that mitigate the effort in exploring alternative solutions for structural implements, and this is the flavor of CautoD that will be considered hereafter.

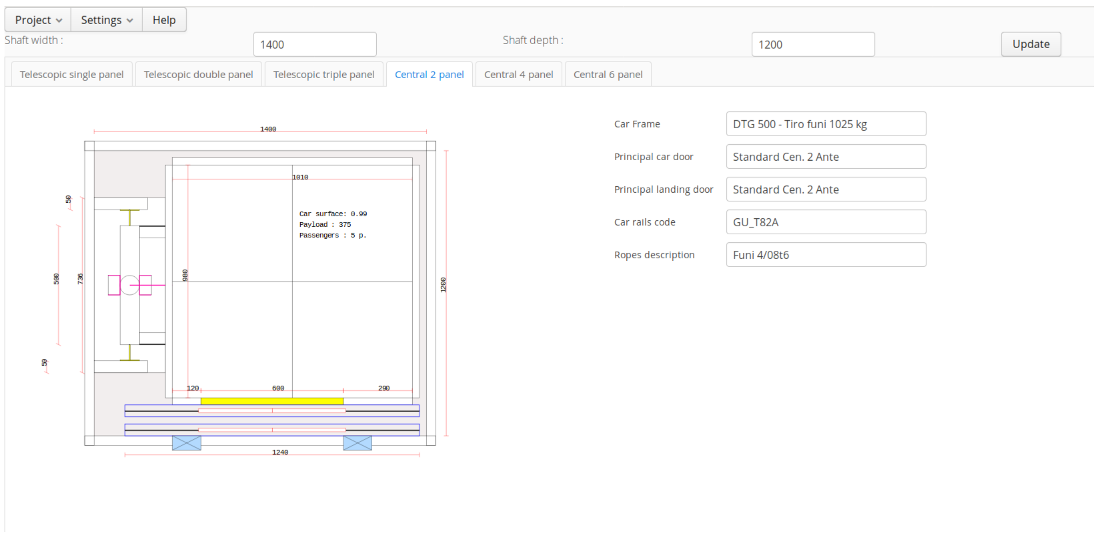
Elevators are complex implements whose design requires the combination of several standard components which must be fitted to custom spatial and usage requirements. Since human designers cannot simulate all possible viable models, they leverage "good design practices", i.e., heuristics, that usually yield reasonable engineering solutions. On the converse, while a program might thoroughly simulate the space of alternative elevator designs, the process is not guaranteed to be computationally feasible. The goal of this paper is to provide experimental evidence to evaluate which approach is best in which circumstances, to serve as guidance for automated modeling of elevator systems. As the experimental results clearly show, the impact of heuristics in pruning the search space of feasible solutions can be dramatic, enabling CautoD programs to achieve a number of good alternative solutions with very little computational effort. Still, a computer-intensive strategy that filters feasible designs using *a-posteriori* reasoning, can find designs that are disregarded by heuristics, but that could still provide useful solution in specific niche conditions.

BACKGROUND AND MOTIVATIONS

With nearly 5 million installations in EU-27 countries as of 2012, elevators are complex automation systems which nevertheless are part of daily routine for hundreds of thousands of non-technical users. It is a fact that most installations are to be found in residential and tertiary buildings, with industrial sites accounting for a mere 4% of the total market share — see [DAHP+12] for more details. In Italy alone, as of 2015, the number of operational elevators was approaching 1 million, with a total market value of about 1.3 Beuro whereof 366 Meuro are coming from new installa-

Leopoldo Annunziata is an independent professional and mechanical engineering consultant for lift builders and contractors. E-mail: `l.annunziata@studio-annunziata.it` — Marco Menapace and Armando Tacchella are with "Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi" (DIBRIS), University of Genoa, Viale Causa 13, 16145 Genoa, Italy. E-mail: `marco.menapace@edu.unige.it`, `armando.tacchella@unige.it`. The corresponding author is Armando Tacchella.

Proceedings 31st European Conference on Modelling and Simulation ©ECMS Zita Zoltay Paprika, Péter Horák, Kata Váradi, Péter Tamás Zwierczyk, Ágnes Vidovics-Dancs, János Péter Rádics (Editors)
ISBN: 978-0-9932440-4-9/ ISBN: 978-0-9932440-5-6 (CD)



Shaft width : 1400

Shaft depth : 1200

Car Frame	DTG 500 - Tiro funi 1025 kg
Principal car door	Standard Cen. 2 Ante
Principal landing door	Standard Cen. 2 Ante
Car rails code	GU_T82A
Ropes description	Funi 4/08t6

```
Car surface: 0.99
Payload : 375
Passengers : 5 p.
```

Fig. 1: Main view of LIFTCREATE under the guideline which maximizes door size given shaft size. The designer can change shaft width and depth in the top bar, and then ask LIFTCREATE to compute feasible solutions with the “Update” button. Alternative designs appear as tabs, each featuring different types of doors in this case. The plan view showed in the picture is about a hydraulic elevator — piston and car frame are visible on the left side of the car — and a central sliding door with two panels — both car ande landing doors are shown on the front side of the car.

tions[1]. With such volumes, intended users and an expected operational life of decades, it is not surprising that elevators are subject to stringent normative requirements which discipline their design, construction, initial test and maintenance. As products, elevators are considered to be in their “maturity” stage, but the approach to their design, construction and maintenance still shows a lot of room for improvements. In particular, while elevator design is mostly a manual process based on “classical” productivity tools such as CAD programs, spreadsheets and word processors, a push towards innovation comes from the trade-off between relatively small profit margins, and the need to perform an accurate design in order to comply to the above mentioned normative framework. In this direction, automating the design of elevators, while it cannot (and should not) replace certified professional designers, may support them to reduce design time and cost without sacrificing the overall quality of the project.

Currently, only two publicly available products are endowed with some CautoD functionality targeted to elevator design: LIFTDESIGN[2] from DigiPara® and ASCENSORI[3] from ApplicativiCAD. Both applications offer libraries of commercial off-the-shelf components wherewith 2D elevator drawings (plan and vertical views) are generated trying to accomodate physical constraints, designers’ choices, and customers’ requirements. While LIFTDESIGN can also generate 3D models, it consists of “predefined elevator parameters, component structure and elevator logic” which makes the creation of customized solutions rather difficult. Furthermore, LIFTDESIGN does not provide guidance to the designer amidst alternative implementations, but it just provides warning and error messages when drawing generation is attempted in the presence of conflicting parameters. ASCENSORI provides more support for customization and more design automation than LIFTDESIGN, in that it guides the user through various steps of the design by trying to ban alternatives that will almost surely lead to unfeasible designs. The main issue with ASCENSORI is that it relies on a rather contrived and acronym-laden graphical interface which, together with some maturity issues, severely affects usage by all but the most experienced designers. Unfortunately, neither of the two applications is available for research purposes, so they cannot be used as a platform for comparative evaluations and implementation of new CautoD techniques to extend existing functionalities.

Providing designers with an easy-to-use, yet flexible tool with full-fledged CautoD functionality for elevators is the main aim of the AILIFT suite[4] Currently in its early prototypical stage, AILIFT is expected to group three different but synergistic applications, namely LIFTCREATE to generate structural designs, LIFTREPORT to generate accompanying documentation for installation and certification of the elevator, and LIFTPLAN to generate detailed parts count. The core CautoD functionalities are implemented by LIFTCREATE which takes the designer from the very first measurements and requirements, e.g., shaft size and payload, to a complete project which guarantees feasibility within a specific normative framework. To achieve this, LIFTCREATE works in two steps. In the first step, the user is asked to enter relevant parameters characterizing the project, and an overall “design philosophy” to be implemented. For instance, if the size of the elevator’s shaft is known and fixed in advance, LIFTCREATE can generate solutions which maximize payload, door size, or car size. A design philosophy is just a set of guidelines which, e.g., prioritize door size over other elements, still keeping into ac-

[1] Mediacom report on the Italian elevator market (2016). Available from: http://www.anacam.it/.

[2] `https://www.digipara.com/products/liftdesigner/`.

[3] `http://www.applicativicad.it/ascensori.php`.

[4] `http://www.ailift.it`.

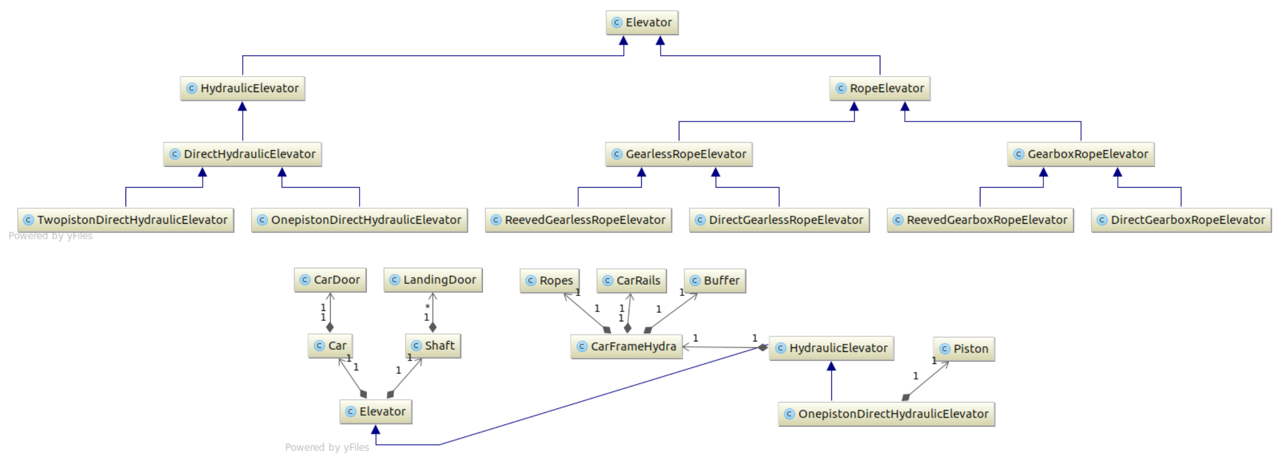


Fig. 2: Taxonomy of elevator types handled by LIFTCREATE (top) and components of OnePistonHydraulicElevator (bottom). Rectangles represent entities, IS-A relations are denoted by solid arrows, and HAS-A relations are denoted by diamond-based arrows.

count hard constraints, e.g., payload and car size should not fall below some threshold. In the second phase, LIFTCREATE retrieves components from a database of parts and explores the space of potential solutions. This is exactly where the tradeoff between heuristic and computer-intensive methods becomes relevant, since the solution space grows as $O(p^s)$, where p is the number of different parts to be fitted and s is the maximum number of alternatives for each part. Since the growth of such space is exponential in s, and the number of alternatives for each part can be large — consider, e.g., different suppliers, different mechanisms, and different builds — it is easy to see that the space of alternative designs is subject to a "curse-of-dimensionality" issue. In principle LIFTCREATE should be able to output optimal solutions, i.e., designs that maximize desiderata within the given constraints. However, for the second phase to be computationally efficient, by default LIFTCREATE aggressively prunes the search space of alternative designs using heuristics. This implies that the set of designs presented to the user in the end could be incomplete and single designs could be sub-optimal. If enough computational resources are available, LIFTCREATE can afford to simulate more designs among which the optimal ones can be picked and presented to the user. However, since this implies less aggressive search space pruning, the computer-intensive approach is to be evaluated carefully to avoid blow-up in computation time.

CASE STUDY

As shown in the taxonomy of Figure 2 (top), elevators can be differentiated in two broad categories, namely traction — also called rope in the following — and hydraulic elevators. In traction elevators, the car is suspended by ropes that are moved via an electrically driven sheave. The opposite end of the ropes is connected to a counterweight. Depending on whether the sheave is driven directly by the electric motor or whether a gearbox is used, these elevators are further differentiated into geared and gearless traction systems. According to [DAHP+12], geared traction elevators are the most common "legacy" elevator type in Europe, constituting more than two thirds of the European elevator stock. Gearless traction elevators are a comparatively young technology and only constitute about 8% of the total elevator stock. The remaining elevators operate on hydraulics, i.e., they rely on one or more pistons to move the car. Energy is usually provided to the hydraulic fluid by an electrically driven pump, and typically no counterweight is needed to compensate for the weight of the car. Hydraulic elevators (HEs) are often used in low-rise applications and are widely used in new installations in some European countries, including Italy: Their low initial costs, compact footprint and ease of installation makes them the most viable choice for retrofitting old residential buildings, and a cost-effective solution for new ones alike. The choice of HEs as a case study is thus motivated by their popularity, and by the fact that, in spite of their relative low part count, their structure presents already most of the challenges that are to be found in other elevator types.

The components of HEs considered by LIFTCREATE CautoD procedures are shown in Figure 2 using an UML class diagram to outline the corresponding part-whole hierarchy. Notice that, in order to manage the space of potential designs components cannot be solely available as drawing elements, like in classical CAD solutions, but they must be handled as first class data inside LIFTCREATE logic. In particular, OnePistonDirectHydraulicElevator is both a leaf entity in the taxonomy shown in Figure 2 (top), and also the root node of corresponding part-whole hierarchy in Figure 2 (bottom). Looking at the hierarchy, the structure of HEs with one piston direct drive can be easily learned, the only peculiar aspect being that these implements feature only one piston (Piston). The remaining components are common to HydraulicElevator or Elevator. In particular, the car frame (CarFrameHydra), i.e., the mechanical assembly connecting the car with the piston, is specific of hydraulic elevators. Albeit not physically part of the car frame, the entities CarRails, i.e., the rails along which the car is constrained to move, Buffer, i.e., the dumping device placed at the bottom of the elevator shaft, and Ropes, are logically part of it since their type and size must be inferred from or melded with the type and size of the car frame. Common to all elevator types, the entities Shaft and Car are both logically part of the Elevator entity, but only Car is also a physical component, together with its sub-component CarDoor. In the case of Shaft, while landing doors (LandingDoor) are not physically part of the shaft, they are attached to it and their size and type must be inferred from or melded with car doors. The relationships encoded in such part-whole hierarchy are instrumental to LIFTCREATE when it comes to handle drawing, storage and retrieval of designs, but also to reason about the various trade-offs of a design when searching in the space of potential solutions, as described in the next section.

AUTOMATED DESIGN METHODOLOGIES

For the sake of clarity, in the ensuing discussion about LIFTCREATE CautoD procedures for hydraulic elevators it is assumed that only one supplier and build are available for car frames — including all logically-attached components, i.e., car rails, buffers and ropes — and for doors. This is not a severe limitation, as often designers and elevator installers will have their preferred pool of suppliers and builds for car frames and doors, opting for different ones only when the setup requires solutions which are manufactured only by specific suppliers. The CautoD procedure operates according to some predefined parameters:

- Reductions, i.e., distances from car to shaft on those sides of the car which are free from doors and car frame.
- Car wall thicknesses (different values for each car wall).
- Maximum car frame overhang (distance from the central axis of the car and piston).
- Choice of reduced or standard landing door frames.
- Door size tolerances with respect to other components, e.g., car frame.

Finally, it is assumed that the car will have only one door on the front, and that the car frame is to be placed either on the left side or at the back of the car. The case in which the car frame is placed to the right is simmetrical to the one considered.

Independently from whether LIFTCREATE uses heuristics or computer-intensive methods to guide the designer amidst alternative choices, the CautoD procedure scheme is the following:

1. Shaft size (width and depth) is input by the designer; no other configuration parameters are necessary since it is assumed that there is only one door on the front side of the car.
2. All available car frames are considered in ascending payload order; each selected car frame is placed either on the left or at the back of the car, aligned to its center.
3. Taking into account the selected car frame size, car wall thicknesses and reductions, the current internal width of the car is computed.
4. Door selection depends on whether heuristics or computer-intensive techniques are used (see below)
5. For each selected car frame and door, the weight of the car — doors included — and its payload are computed; given also the maximum overhang, it is possible to validate the selected car frame: if adequate, the current solution is saved into a list of feasible designs; otherwise, the solution is discarded and the procedure goes back to step (3).

To complete point (4) in the procedure scheme above, one could resort to either heuristics or computer intensive methods. In the former case, the following steps are taken:

a. The "internal cabin door" parameter — ICD in the following — is computed starting from the value computed in step (3) above, considering the size of landing door frames.

b. In order to select car and landing doors of feasible size, the ICD parameter, the shaft width, and the door size tolerances are considered to perform checks depending on the door types, i.e., sliding and folding; for the sake of brevity, details are omitted, but it is important to notice that such checks involve some non-trivial reasoning about door placement.

c. The doors selected at step (b), together with the selected car frame are part of the evaluation carried out in step (5) of the CautoD procedure scheme.

If computer-intensive methods are opted for, the following steps are taken:

a. Door sizes larger than the shaft are filtered away.

b. For each combination of door size and selected car frame, the parameter "residual car space" — RCD in the following — is computed; the parameter amounts to the difference between the shaft width and the total space allocated for the door; RCD is computed for each door type, since the space available for door placement is clearly a function of the current combination of door, car-frame, reductions and car wall thicknesses.

c. RCD is divided up into intervals of equal length — 5mm in the current implementation — so that a set of projects is generated, each with a different door placement; some of these projects will not be feasible, because door placement will not be coherent with the overall constraints, but these will be filtered at the end of the CautoD procedure.

While heuristics generate projects that are guaranteed to be feasible, the computer-intensive approach requires post processing to filter out remaining unfeasible projects. There are five checks that serve this purpose, namely:

- the door should not be placed outside the shaft;
- the door opening should be contained in the car;
- the car frame should be contained in the shaft;
- there should be no interferences between car doors and car frame;
- the landing door frame, once aligned to the car door, should not be outside the shaft;

If at least one of the checks above fails, the corresponding design is discarded.

EXPERIMENTAL EVALUATION

The experimental evaluation is carried out considering eleven hydraulic elevator case studies — CS in the following. These include both configurations for which there exists feasible solutions, and configuration for which there are none. Among configurations for which feasible designs exist, both typical and "borderline" cases are considered. In more details:

- CS#1 features a shaft size which is too small to have feasible solutions;

- CS#2 is the minimum shaft size to have exactly one feasible solution: clearly the solution found has to be the same across heuristics and computer-intensive methods;
- CS#3-7 represent "typical" shaft sizes found in residential buildings;
- CS#8-10 feature unconventionally large shaft sizes;
- CS#11 features a shaft size which is too large to have feasible solutions.

In all the cases above, feasibility is constrained by the working hypothesis outlined before and by the available set of components. For instance, in CS#11 there are no feasible designs because in the component library there are no car frames available that can handle the resulting maximum payload. The experimental results herewith presented are obtained using LIFTCREATE prototype implemented in Java 8 and based on the SPRING object-persistence framework [5] using Vaadin[6] to generate and display GUIs. To handle components data and generated projects, a local instance of Mysql server 5.7 is adopted. All the simulations are executed on a machine with an Intel i7 5th generation 8 core CPU, featuring 8GB of RAM and running Ubuntu Linux 16.10.

TABLE I: Heuristics vs. computer-intensive methods.

CS	W	D	Heuristics		Computer-Intensive		
			CPU	OK	CPU	GEN	OK
1	900	830	235	0	1504	50	0
2	910	830	147	1	1488	120	1
3	1200	1000	670	122	3260	18972	1777
4	1200	1200	596	174	3736	18078	4181
5	1400	1000	1314	265	4122	40930	5939
6	1400	1200	943	291	3367	36440	11355
7	1600	1000	1606	406	3913	66717	9035
8	1600	1200	1349	470	2801	63854	24935
9	1800	1200	1614	516	2576	73046	28920
10	2000	1200	3628	360	1838	58888	22131
11	2000	1400	4236	0	1266	0	0

Table I presents experimental data about the comparison between heuristics and computer-intensive methods on the selected case studies. In the Table, **CS** is the unique case study id, **W** and **D** are the corresponding shaft width and depth, respectively; both for heuristics and computer-intensive methods, **CPU** is the amount of time (in milliseconds) required to generate solutions, and **OK** is the number of feasible solutions found; **GEN** is the number of generated projects which, in the case of computer-intensive methods, does not readily correspond to feasible solutions, i.e., those that pass the checks mentioned at the end of the previous section. From Table I it can be observed that heuristics and computer-intensive methods do not present substantial differences when it comes to over-constrained configurations. In particular, for CS#1 and CS#2, the only noticeable element is that heuristics are faster than computer-intensive methods, as they can prune many unfeasible designs in the early stages of search — in CS#2 there is a $10\times$ factor between the two. As far as "typical" configurations are considered, the picture changes. The gap in performances is never greater than a $6\times$ factor, and computer-intensive methods are generating a strict superset of the projects generated by heuristics. The difference set is populated by solutions that, albeit feasible, do not correspond to straightforward designer choices, whose spirit is embedded into heuristics methods. Nevertheless, many such designs do have practical value. For instance, since computer-intensive methods explore many alternative door placements, they find solutions which often end up being preferred by implementors because they allow an easier fitting of cables or other implements, whereas customers may prefer them, e.g., because of aesthetic reasons. Finally, as for configurations which admit many alternative solutions, it can be observed that both heuristics and computer-intensive methods struggle with an ever-increasing search space. In some cases, e.g., CS#10, *a-posteriori* pruning techniques implemented in the computer-intensive approach end up being more efficient that heuristics.

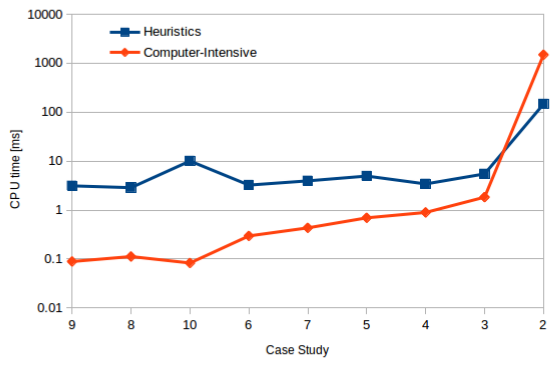


Fig. 3: Average time to compute a single solution; case studies appear along the x-axis, ordered descendingly according to their estimated complexity; y-axis report CPU times in milliseconds, on a logarithmic scale; the plot report the performances of heuristic (squares) vs. computer-intensive (diamonds) methods.

In figure 3 an alternative view of the results shown in Table I is shown. In the Figure, the average time to compute a single solution for heuristics and computer-intensive methods is plotted. For each case study, the average time per solution is just the ratio between the total time and the number of feasible projects — columns CPU and OK in Table I. However, case studies are sorted along the abscissa of the plot considering an ascending order of configuration complexity, whose approximate indication can be obtained considering the number of solutions generated by the computer-intensive approach — column GEN in Table I. The principle behind this choice is that, the more solutions are evaluated by computer-intensive techniques, the less constrained the original configuration is, and the less complex the overall problem is. Notice that CS#1, CS#2 and CS#11 are excluded from the plot in Figure 3 since the corresponding data would not make any sense. What can be observed from the plot is that, when the complexity of the problem is low, computer-intensive methods may have an edge over heuristics: it takes more than $10\times$ time to compute a single solution using heuristics, on average. The gap becomes

[5]`https://spring.io/.`
[6]`https://vaadin.com/home.`

smaller and smaller while the complexity increases. This is to be expected, because computer-intensive methods will have to reject more and more solutions, increasing the overhead to reach feasible solutions. For the most complex design, namely CS#2, the cost of computing 120 solutions in order to reach just one, is overwhelming with respect to what can be achieved using heuristics. This reveals that, all other things being equal, computer intensive methods have an advantage over heuristics when the problem complexity is relatively low, whereas contrived configurations might benefit the most from the pruning power of heuristics.

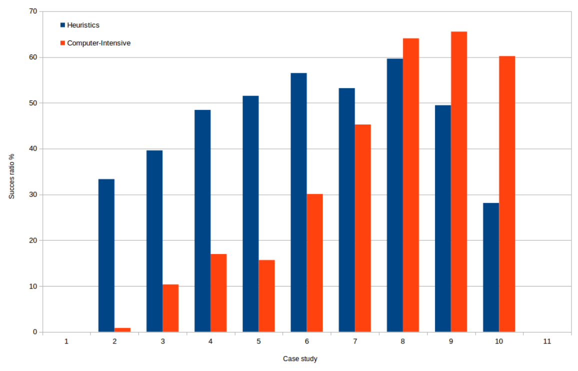


Fig. 4: Success rate (percentage); each bar represents the success rate of either heuristic (blue) or computer intensive (orange) methods.

One last perspective about the results is shown in Figure 4. Here the success ratio of heuristics and computer-intensive methods is computed as a percentage value, considering the number of feasible designs and the number of attempts made to generate them. While in the case of computer-intensive methods this is simply the ratio between columns OK and GEN in Table I, in the case of heuristics the value is computed considering the number of solutions that are "early-pruned", i.e., those which heuristics do not attempt to extend into a full-fledged solution. For all the case studies, except CS#1 and CS#11 which do not admit feasible solutions, Figure 4 tells that the success ratio of heuristics falls below 30% only in one case, i.e., CS#10, whereas computer-intensive methods are more configuration-dependent. In particular, the success ratios of "easy" configurations — namely CS#8-10 — are much higher than relativealy "hard" configurations, e.g., CS#2, but also CS#3-5. This confirms the evidence exctracted from the plot in Figure 3, i.e., that heuristics have an edge over computer-intensive methods on "hard" configurations, whereas "easy" configurations can be within the reach of computer-intensive methods.

RELATED WORK

Searching the literature for contributions related to CautoD, one of the most recent contribution to be found is Bye et al. paper [BOP$^+$16] on automated design . Here the authors focus on the design of offshore cranes and, more specifically on the optimization of parameters related to the design. The spirit of the contribution is very similar to the one given herein, but the domain and the techniques considered are quite different. The same applies also to [RPL$^+$16], wherein the exterior lighting is subject of the CautoD procedure.

Other recent related works include [WRWS15], where a cloud-based environment is proposed to support design and manufacturing. This contribution is somewhet related in the sense that leveraging a cloud deployment is an expected development in order to achieve improvements in the design process, e.g., by pooling solutions and allowing designers to browse solutions by problem similarity. In [CRS$^+$13], the problem of knwoledge representation is considered in automated designed is considered. Since AILIFT uses an ontology of components to represent the elevator structure as well as the relations between the design componets, it falls in the taxonomy of potential representations outlined in that paper. While [CRS$^+$13] considers many other alternatives to (formal) ontologies, the investigation of potential improvements in this direction is left for future research.

CONCLUSIONS

Summing up, this paper compares heuristics and computer-intensive methods in the automated design of hydraulic elevators. The main lessons learned from our experimental evaluation are the following:

- Heuristics are comparatively faster than computer-intensive methods on contrived configurations and generate a manageable number of solutions among which the preferred one can be picked even manually; however, they do miss "good" practical solutions that can be found by computer intensive methods; this makes heuristics a good default choice in a CautoD procedure whose main purpose is to provide feasibility assessement and budget estimates, run by personnel in non-technical businness units, e.g., sales department.
- Computer-intensive methods are able to find feasible solutions that are missed by heuristics with an increase in CPU time which is not overkilling in most cases; while the number of generated solutions cannot be managed manually, suitable helper procedures can be implemented to allow designers to navigate among the solutions in order to perform design fine tuning for niche configurations.

Considering future extensions and, more specifically, elevators with more than one door, it can be observed that the number of feasible solutions given n feasible single door placements grows as $O(n^2)$. In this case the interplay between heuristics and computer-intensive methods can address the problem of generating enough configurations to be analyzed, yet prune rapidly those that would not lead to feasible designs and would therefore clutter the efficiency of the overall process. These issues are even more compelling when considering a number of different componenent suppliers. Indeed, while the experimental analisys herewith presented is limited to only one set of components from one supplier, common practice involves mixing and matching components from several suppliers in order to meet structural as well as economical constraints. The implementation of the full AILIFT suite, including a complete database of components and suppliers, will enable furthering the analysis along these lines.

REFERENCES

[BOP+16] Robin T. Bye, Ottar L. Osen, Birger Skogeng Pedersen, Ibrahim A. Hameed, and Hans Georg Schaathun. A software framework for intelligent computer-automated product design. In *30th European Conference on Modelling and Simulation, ECMS 2016, Regensburg, Germany, May 31 - June 3, 2016, Proceedings.*, pages 534–543, 2016.

[CRS+13] Senthil K Chandrasegaran, Karthik Ramani, Ram D Sriram, Imré Horváth, Alain Bernard, Ramy F Harik, and Wei Gao. The evolution, challenges, and future of knowledge representation in product design systems. *Computer-aided design*, 45(2):204–228, 2013.

[DAHP+12] Aníbal De Almeida, Simon Hirzel, Carlos Patrão, João Fong, and Elisabeth Dütschke. Energy-efficient elevators and escalators in europe: An analysis of energy efficiency potentials and policy measures. *Energy and Buildings*, 47:151–158, 2012.

[KL63] Louis A. Kamentsky and Chao-Ning Liu. Computer-automated design of multifont print recognition logic. *IBM Journal of Research and Development*, 7(1):2–13, 1963.

[Pea84] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[RPL+16] Hugo Rocha, Igor S Peretta, Gerson Flávio M Lima, Leonardo G Marques, and Keiji Yamanaka. Exterior lighting computer-automated design based on multi-criteria parallel evolutionary algorithm: optimized designs for illumination quality and energy efficiency. *Expert Systems With Applications*, 45:208–222, 2016.

[RS12] R Venkata Rao and Vimal J Savsani. *Mechanical design optimization using advanced optimization techniques*. Springer Science & Business Media, 2012.

[WRWS15] Dazhong Wu, David W Rosen, Lihui Wang, and Dirk Schaefer. Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation. *Computer-Aided Design*, 59:1–14, 2015.

[ZWPG03] F Zorriassatine, C Wykes, R Parkin, and N Gindy. A survey of virtual prototyping techniques for mechanical product development. *Proceedings of the institution of mechanical engineers, Part B: Journal of engineering manufacture*, 217(4):513–530, 2003.

LEOPOLDO ANNUNZIATA graduated from the University of Genoa, Italy with a "Laurea" (MSc equivalent) in Mechanical Engineering in 1995. He is an independent professional and mechanical engineering consultant working for elevator industries and building contractors. He designed about 500 working elevators in more than 10 years of activity in the field, and he is actively involved in the development of AILIFT-LIFTCREATE prototype since its early development phases.

MARCO MENAPACE graduated from the University of Genoa, Italy with a "Laurea" (MSc equivalent) in Computer Engineering in March 2016. He is now a Research Engineer at the Department of Informatics, Bioengineering, Robotics and System Engineering. He is currently the main developer of the AILIFT-LIFTCREATE prototype with an expertise in Java-based OOD/OOP and techniques for cloud and web-based application deployment.

ARMANDO TACCHELLA graduated from the University of Genoa, Italy with a "Laurea" (MSc equivalent) in Computer Engineering and a PhD in Computer Science and Engineering. Dr. Tacchella was a research associate at Rice University in Houston (US) and then a research fellow at University of Genoa, where he became associate professor of information processing systems in 2005. His research interests are in the field of artificial intelligence and formal methods applied to system engineering.