A Heuristic Approach to the Cable Routing Problem in Electrical Panels

Alexandre Erwin ITTNER [a,1], Claudio Cesar de SÁ [b,2], and Fernando Deeke SASSE [c,3]

[a] *WEG Automação S.A., Departamento de Projetos, Engenharia e Automação, 89256-900 Jaraguá do Sul, SC, Brazil*

[b] *Universidade do Estado de Santa Catarina, Departamento de Ciência da Computação, UDESC, 89223-100 Joinville, SC, Brazil*

[c] *Universidade do Estado de Santa Catarina, Departamento de Matemática, UDESC, 89223-100 Joinville, SC, Brazil*

Abstract In this paper, we present new results concerning the heuristic optimization of cable routing in electrical panels. The problem is modeled and a heuristic solution, using an insertion algorithm and a modified version of the Dijkstra's algorithm, is proposed, analyzed, and compared with human-made solutions. Tests have shown that good results can be obtained from layouts commonly found in the industry.

Keywords. cable routing, electrical systems design, optimization

1. Introduction

In industrial facilities, like hydroelectric power plants, automobile manufacturing plants, and other facilities equipped with medium or large-sized industrial automation systems, there are panels and cabinets using dozens or hundreds of electrical components like contactors, relays, frequency inverters and PLCs. In major installations, these components are wired by thousands of cables, passing through hundreds of conduits arranged as a graph. The arrangement of these cables has direct influence over the cost of the installation and in the design quality. The optimal definition of the routes used by the cables also allows the definition of the quantity and type of the cables used in the panel during the design time.

The problem of giving a path for each cable in the panel, connecting all the associated components at a minimum cost and without overfilling the space available in the conduits, will be called here *the cable routing problem in electrical panels*. This is a real NP-complete problem from the industry, with a large scope, since many other subproblems are embedded in it.

[1] E-mail: aittner@gmail.com
[2] E-mail: claudio@joinville.udesc.br
[3] E-mail: fsasse@alumni.uwaterloo.ca

The routing can be done manaully: empirically or aided by measure systems (a ruler on a scaled drawing or a measurement tool in a CAD application), the designer selects the shortest path for every cable, starting from the most expensive and going to the cheaper ones. If the shortest route between two components becomes overfilled, the designer reroutes the cable through the shortest underfilled route available. If a feasible route cannot be found, the designer moves the already routed cables to another path. This process follows iteratively until all cables are routed.

This is a stressing, repetitive, and error-prone process. This paper analyzes the properties of cable laying in electrical panels and suggest a computer solution that near-optimal solutions for a simplified version of the problem.

2. Modeling

A panel is composed of an arbitrary number of components (contactors, overload relays, fuses, PLCs, etc.), each one with an arbitrary number of connection terminals, in a given physical position, and a set of conduits for wire disposition. Therefore, the model of a panel shows the connection terminals, the conduits and associations among them. Mathematically, the panel, P_n can be represented by

$$P_n := (L_t, L_c, L_i)\,, \tag{1}$$

where L_t denotes the list of connection terminals, L_c denotes the list of conduits, and L_i denotes the list of connections. The designer gives the physical position of the components and terminals, according to design standards, and the routing process is not allowed to change it. The layout of a simplified panel is shown in Figure 1.

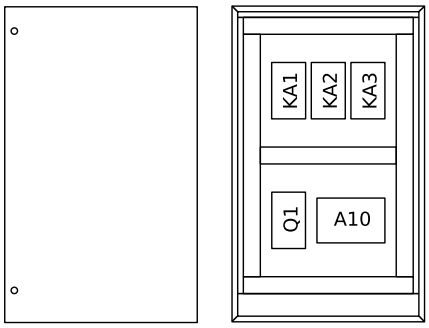

Figure 1. A typical panel

A connection I_n is a set of electrically equivalent terminals that must be connected by cables and is represented by

$$I_n := (L_t, e_i, s_e, c_n)\,, \tag{2}$$

where L_t denotes the list of connected terminals, e_i represents the electrical properties of the cable used for this connection (gauge, color, insulation voltage, maximum allowed

temperature, etc.), s_e is the external cross-section of the cable, and c denotes the cost of the cable. The values of e_i are important regarding the electrical design, but are not used in the routing process. A connection among three terminals is represented in Figure 2.

Each connection gives origin to one or more cables, needed to electrically connect the terminals. A cable w is represented by

$$w_n := (t_i, t_f, e_i, s_e, c)\,, \tag{3}$$

where t_i and t_f denote the starting and ending terminals, respectively, e_i represents the electrical properties of the cable, s_e its external cross-section, and c its cost.

Each cable from the same connection wires two terminals, and no terminal may be connected to more than two cables[4]. Therefore, $q_{term} - 1$ cables are needed to connect q_{term} terminals.

The connection order of the terminals from a connection list is particularly important because it allows some minimization on the distance traveled by the cables. From the combinatorial analysis, it can be shown that there are $q_{term}!$ permutations for the list of terminals and that half of these permutations are inversions of those already enumerated ones. As permutations with inverted connection order of the terminals are not electrically distinct, there are $q_{term}!/2$ ways to sort the terminals of each connection. This also means that the number of available sequences increases exponentially with the number of connected terminals.

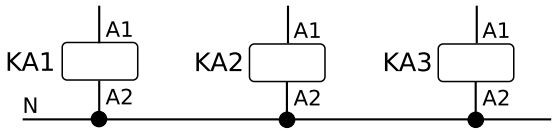

Figure 2. Connection among three terminals of three distinct components

A terminal is represented by

$$T_n := (n_c, n_t, P_{xyz})\,, \tag{4}$$

where n_c is the component name (eg. KA1, KA2, and KA3 from Figure 2), n_t is the identification of the connected terminal (eg. A1 and A2), and P_{xyz} denotes the three-dimensional point with coordinates (x, y, z) of the terminal within the panel space.

Conduits are line segments representing wire ducts, raceways, cable trays, and other materials used to hold cables in electrical panels. A conduit is represented by

$$C_n := (s_c, A_{xyz}, B_{xyz}, t)\,, \tag{5}$$

where s_c denotes the cross-sectional area available for the cables, including safety margins, A_{xyz} and B_{xyz} respectively denotes the starting and ending points of the conduit in the panel space and t specifies the conduit type. The length of a conduit is given by the Euclidean distance between the points A_{xyz} and B_{xyz}. For modeling, there are two types of conduits:

[4]This is valid for terminals connected in a daisy chain. In other configurations, like those in distribution bars, more than two cables may exist.

Open conduits. Conduits that allows the crossing of cables through its walls, as open-slot wire ducts and some types of raceways. An example of this type of conduit is given with dashed lines in Figure 3. Open-type conduits may also be used to model "virtual conduits", as harnesses bound with cable ties.

Closed conduits. Conduits that does not allow crossing, as solid-slot wire ducts, internal raceways and liquid tight conduits. An example of this type of conduit is given with full lines in Figure 3.

The first and the last jump of each cable may pass through the walls of an open conduit if there is no ending nearer to it. The points where the cable crosses the conduit wall are called *entry points* (points $P1$ and $P2$ in Figure 3). These jumps must be considered in the calculation of the cable length and the conduit saturation. The entry points can be determined by solving the following problem: *given a line segment $\overline{AB}$ representing the conduit and a point C representing the terminal, find the entry point D over $\overline{AB}$ for that the length of the line segment $\overline{CD}$ is minimal.* This step is executed for each conduit, so, it will find the nearest entry point to the terminal. Strategies to deal successfully with these entry points are described in Section 5.

The conduit set may be represented as a weighted graph with edges connecting nodes attributed arbitrarily, preserving the topology of the panel, as shown in Figure 3. Conduits have a limited available internal space, so the amount of cables transiting through an edge is limited by the sum of their cross section areas. A conduit is called *overfilled* if it cannot hold more cables due to this limitation.

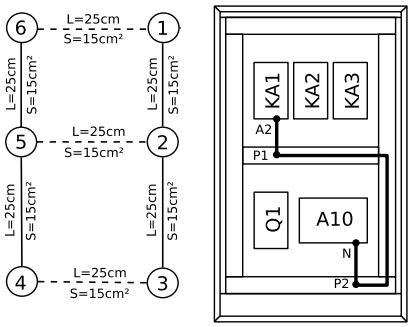

Figure 3. Conduit graph of the panel from Figure 1 and routed cable

A route is, by definition, a sequence of nodes and terminals that gives the path followed by a cable from the starting terminal t_o and the ending terminal t_d, i. e.,

$$r_n := [t_o, n_1, n_2, \ldots, n_n, t_d]\,, \tag{6}$$

where n are the nodes traveled by the cable. If the cable passes through *open* conduits, the entry points must be listed too.

The length $len(r_n)$ of a route r_n is given by the sum of the lengths of the conduits traveled by the cable, including any entry point, i.e.,

$$len(r_n) := \sum_{i=1}^{i=m-1} dist(n_i, n_{i+1}) \tag{7}$$

where $dist(n_i, n_{i+1})$ is the Euclidean distance between a n_i and n_{i+1}.

Formally, the *cable routing problem* is the problem of finding a set of cables $L_w = \{w_1, \ldots, w_m\}$ and a set of routes L_r for each connection i_n from the list of connections, so that the function

$$c_{total} := \sum_{j=1}^{j=n} c_{unit}(i_j) \times \sum_{k=1}^{k=m} len(R(w_k)) \tag{8}$$

is globally minimal. Here $c_{unit}(i_j)$ is the cost per unit of length of the cable used for the connection i_j, and $R(w)$ is the function that links a cable w to a route from the set of all possible routes.

Given the set of conduits L_c of the panel and a set of cables L_w passing through a conduit $c \in L_c$, the function $R(w)$ must satisfy the constraint

$$\sum sect(w) \leq s_c \forall w \in L_w, c \in L_c\,, \tag{9}$$

where $sect(w)$ is the external cross section area of the cable w and s_c the cross section area available in the conduit c.

3. Computational Complexity

The cable routing problem in electrical panels can be divided as a three-level optimization problem, since three major steps are needed for finding the optimal solution: (a) Given the connections among some terminals, generate a set of cables connecting them; (b) Route these cables through the shortest paths without violating the constraints in the conduits; (c) Sort these routes for minimal cost.

Routing one cable through the shortest path of a non-constrained graph is trivial and done in polynomial time with the Dijkstra algorithm. But this approach only finds the shortest path between two terminals and does not minimize the path of a set of cables needed to connect three or more terminals. Adding this requirement unfolds the problem into a combinatory optimization problem.

Adding the cost and constraints on conduit usage gives to the problem some similarity to the classical, NP-complete[1], *knapsack problem*: there are a bag of limited size (the set of conduits) and a set of objects (cables), with distinct costs, that must be inserted optimally in the available space. However, the cable routing problem has one more degree of complexity, since the cost of a cable changes as the problem evolves and the conduits become saturated.

4. Previous Work

No work featuring exactly the same requirements as the problem described in Section 2 was found in the literature, but several problems with similar requirements were found.

Kloske and Smith [2] presented a solution to a cable routing and optimization problem using genetic algorithms. The problem described features cost minimization, raceway overfill, cable weight, and voltage drop. Unlike the cable routing problem in electrical panels, the problem presented does not have neither open conduits nor connections

with more than two terminals. Ma *et al* [3] proposed a two-level genetic algorithm with two-level chromosome coding for a cable route optimization problem, combining route search and route combination into a hierarchical genetic algorithm.

The cable routing problem stated in Section 2 has several similarities with the problem of finding pathways in biochemical metabolic networks [4,5]. Dooms *et al* [6] introduced the graph-domain constraint programming (CP) and used this strategy to find that pathways. Viegas e Azevedo [7] repeated these results using constraint logic programming (CLP) with a new declarative, ECLiPSe-based, framework called GRASPER. These similarities and results suggest that graph-domain constraint logic programming may also be used to solve the cable routing problem in electrical panels.

5. Handling of Open Conduits and Partial Saturation

The presence of open conduits in panels raises some considerations regarding to the *partial saturation* of a conduit, that happens when some segment of a conduit becomes saturated, but other segments of the same conduit still having enough space for more cables. Figure 4 shows an example of a partially saturated conduit.

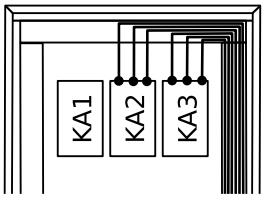

Figure 4. Partial saturation of a conduit.

This problem may be addressed by adding a process called *conduit splitting* that transforms all open conduits in a set of closed conduits delimited by the entry-points of the cables crossing it. Several strategies may be devised for the conduit splitting, but three of them have practical significance:

a. Nearest entry point. The conduits are splitted exactly over the entry point, as shown in Figure 5a. This strategy is simple and gives the more precise results regarding to cable lengths, but may generate too many short edges, increasing the complexity of the graph and the solving time, without much improvements in the results.

b. Nearest entry point with approximations. This strategy avoids creating short edges by finding the entry point and searching for conduits ending nearer than a user-specified threshold distance td from that entry point. If there is such conduit, no splitting is performed and the cable enters the conduit by the existing ending, as shown in Figure 5b. This strategy gives good precision, but is slower than the former, since it also requires a search for conduit endings.

c. Constant length splitting. This strategy splits the conduits in a regular distance sd, as show in Figure 5c. It is a fast strategy, since it does not need searches for existing edges nor nearest point calculations, but it may create useless conduit segments, i.e. conduit segments that will never be used because there is no near terminal.

Strategies *b* and *c* also allow the user to select the threshold distances and, therefore, control how much effort will be applied to the optimization of the routes. Threshold and splitting distances may also be heuristically selected according to the size of panel and properties of the circuit (i.e. shorter distances for small, heavily wired panels and longer distances for larger and sparsely wired ones).

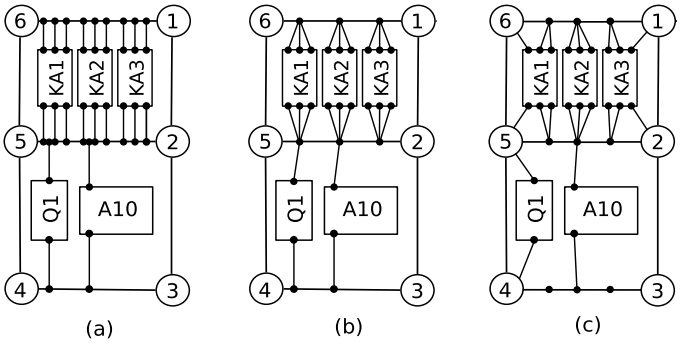

Figure 5. Three strategies for conduit splitting: (a) nearest entry point, (b) nearest entry point with approximations, and (c) constant length splitting.

6. Simplification

The model described in Section 2 allows high-quality solutions, but, its higher complexity inspires the search for a simplified model that allows fast near-optimal computer solutions. Therefore, the routing process was applied on a defined list of cables, not on a list of connections. This simplification cuts the computing complexity, but has the drawback of excluding the search for better solutions by changing the terminal wiring order. So, a cable will follow the shortest route between two terminals unless this route precludes, by conduit saturation, the existence of more economic routes for other cables.

7. Algorithm

The proposed algorithm (see Algorithm 1) runs in two steps: First, it performs the conduit splitting, using the "constant length splitting" strategy presented in Section 5 to convert all open conduits into closed ones. In the second step, it iterates through the list of cables, inserting them cables into the panel in descending order of cost, routing them through the shortest path, and decrementing the available space according to the cross-section area of the cable (this last process is called *graph shrinking*).

The cost minimization comes from the sorting, by inserting first the most expensive cables and giving them the shortest routes. The cheaper cables are routed later, getting increasingly worst routes due to conduit saturation.

The algorithm works with permutations to satisfy the saturation of cables in a specific conduit. This process is similar to the backtracking process available in languages as Prolog [8,9]. If, due to route saturation, a cable cannot be inserted, the current solution is

discarded, the list of cables is permuted and the process begins again. If no permutation yields success, the problem is said to be non-feasible.

Algorithm 1 Cable routing

Given the lists of cables L_w, terminals L_t, conduits L_c, nodes L_n, and a minimum conduit length l_{min};
For each open conduit $c \in L_c$ with $length(c) > l_{min}$, do:
    Create a set of closed conduits L'_c so $length(c') \leq l_{min} \forall c' \in L'_c$ and the concatenation of $[L'_{c_1} \dots L'_{c_n}] = c$ and inserts it in L_c;
    Remove c from L_c;
Sort L_w, in descending order, according to the cable cost per length unit;
While no valid solution is found, do:
    For each cable $w \in L_w$, do:
        Find the starting and ending nodes in the graph, for that $dist(w_{start}, n_{start})$ and $dist(w_{end}, n_{end})$ are minimal;
        Find the set L'_c containing the conduits of L_c with enough internal space for the cable w;
        Find the shortest path r_w between nodes n_{start} and n_{end} of L'_c;
        Update the available space in the conduits of L_w according to r_w;
        Find the route cost $c_{route} = c_{unit} \times (dist(w_{start}, n_{end}) + len(r_w) + dist(w_{start}, n_{end}))$
    Find the total cost from the current solution;
    End the program if a feasible solution was found; Otherwise, permute L_w;

According to the insertion heuristics, it is expected that a solution can be found without the need of too many permutations. A solution is considered optimal if no cable was shifted from its shortest paths due to conduit saturation.

8. Implementation

The Algorithm 1 was implemented in the Lua programming language [10,11]. In this implementation, data files with information on the panel geometry and the wiring list are loaded by the application using the language's own parser, run through a validation routine, and used to build the adjacency matrix used by the variant of Dijkstra's Algorithm and tables of nodes, terminals and positions. Lua coroutines are used to generate permutations for the list of cables. An OpenGL viewer was also built to ease the validation of the models.

The implementation of the Dijkstra's Algorithm [12] used to find the shortest path between nodes differs from the standard algorithm by considering only conduits with enough space for the cables. Therefore, only the adjacency matrix is needed for each instance, lowering the need for processing. In order to allow the search on non-directed graphs, as the graph of conduits, the adjacency matrix keeps two references for each conduit.

8.1. Tests and Results

A case-study was performed comparing the solution generated by the implementation described in 8 with a solution given by a human expert, at the time of panel assembly, using his professional knowledge but without any formal procedure. It is the project[5] of a PLC remote panel with 57 segments of conduit and 692 wires and cables connecting 1096 distinct terminals.

The Figure 6 shows the physical layout of the components. This panel was chosen because there is a multitude of alternative routes and a high concentration of wires in the area near to the PLC, allowing to test the main characteristics of the algorithm. The data was originally generated from the wiring list used for the panel assembling and from a three-dimensional model, made with a CAD application, that gives the physical position of each terminal and conduit.

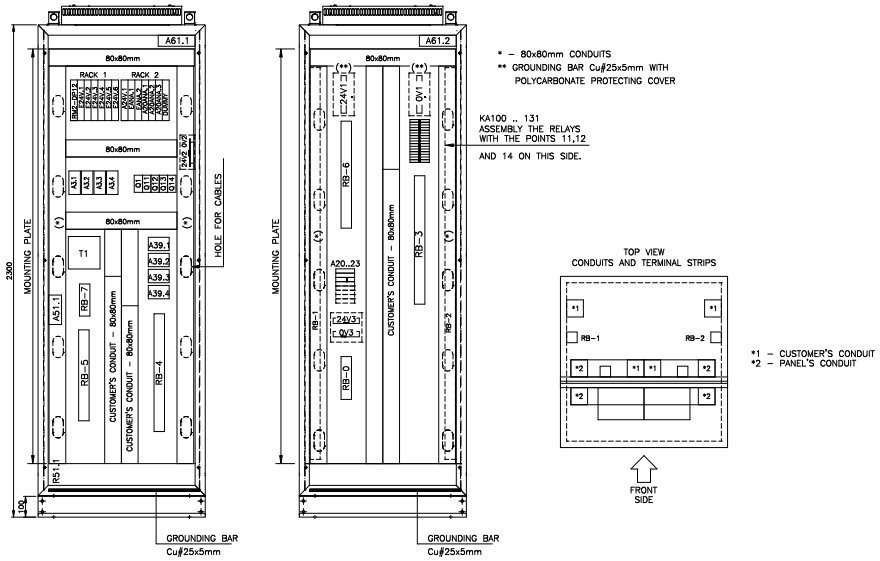

Figure 6. Panel from the case study (Source: WEG Automação S.A.)

The test consists of running the program with the conduit splitting parameter set for: no splitting, 300mm, 200mm, 100mm, 50mm and 10mm. In all the tests, all cables are routed through the panel in the first iteration of the algorithm. These tests were performed in a Core 2 Duo 1.8GHz computer with 2GiB of RAM running Linux and the LuaJIT 1.1.3 just-in-time compiler. To minimize timing errors, the program was run five times for each setting and the times presented are the average of these runs, measured with the Unix `time` command, and includes the time needed to start the interpreter, compile Lua code into machine code and load the data files.

Table 1 shows the number of nodes and conduits after the graph splitting and the corresponding running times. Table 2 shows the amount of cables calculated for each test and the amounts bought, at time of panel assembly, with assistance of the human expert

[5]Project number 035815E/05, October of 2005, courtesy of WEG Automação S.A.

using his professional knowledge, but without any formal procedure. Costs are given in Brazilian Reais.

This test shows that in all instances of shielded cable, the most expensive one, got good routes, causing substantial savings. The amount of the cheaper $0.75\mathrm{mm}^2$ dark blue cable given by the algorithm was higher than the amount given by the expert. Also, it must be noted that the expert rounded up the amounts that could not be safely calculated (the $4.0\mathrm{mm}^2$ green/yellow cable is an extreme case). The focus of the algorithm on the cost minimization may be inferred from the global cost decrease. The test also shows that lowering the minimum conduit length leads to more precise solutions, but also increases the running times exponentially.

Table 1. Running times

Splitting	Conduits	Nodes	Running time
No splitting	57	46	0.54s
300mm	77	66	1.01s
200mm	94	83	1.51s
100mm	159	148	4.69s
50mm	299	288	17.88s
10mm	1348	1337	432.52s

9. Concluding Remarks and Future Work

The proposed algorithm gives good results when solving an instance of the problem that features the most common routing requirements, when the space available in the conduits is not tightly restricted. Solutions using the strategy for handling open conduits give more precise results than those using only closed conduits and it also allows the selection of the desired precision level by setting the minimum conduit length. As expected, the running times increases exponentially according to the selected precision level.

Future work involves a comparison between this approach with that one using Genetic Algorithms. It is worth mentioning that a comparison with [2] and [3] is not appropriate here, since in these papers the problems have no entry points. Another interesting development would be the application of graph-domain constraint logic programming in the generation of cable lists from the connection list. This would allow the implementation of the complete model described in Section 2.

References

[1] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice Hall, 1982. PAP ch 82:1 2.Ex.

[2] D. A. Kloske and R. E. Smith. Bulk cable routing using genetic algorithms. TCGA Report No. 94001, University of Alabama, Tuscaloosa, 1994.

[3] Xuan Ma, Kazuhiro Iida, Mengchun Xie, Junji Nishino, Tomohiro Odaka, and Hisakazu Ogura. A genetic algorithm for the optimization of cable routing. *Systems and Computers in Japan*, 37(7):61–71, 2006.

[4] Yves Deville, David Gilbert, Jacques van Helden, and Shoshana Wodak. An overview of data models for the analysis of biochemical pathways. *Briefings in Bioinformatics*, 4(3):246–259, September 2003.

[5] Gregoire Dooms, Yves Deville, and Pierre Dupont. Constrained metabolic network analysis: discovering pathways using CP(Graph). In *Proceedings of the Workshop on Constraint Based Methods for Bioinformatics*, Sitges, Barcelona, Spain, October 2005.

[6] Gregoire Dooms, Yves Deville, and Pierre Dupont. Cp(graph): Introducing a graph computation domain in constraint programming. In *11th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, No. 3709*, pages 211–225, Sitges, Barcelona, Spain, 2005. Springer-Verlag.

[7] Ruben Viegas and Francisco Azevedo. GRASPER: A Framework for Graph Constraint Satisfaction Problems. December 2007.

[8] Yoav Shoham. *Artificial Intelligence Techniques in Prolog*. Morgan Kaufmann Publishers, Inc, San Francisco, 5th edition, 1994. ISBN 1-55860-319-0 (cloth).

[9] L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, Cambridge, Massachusetts, 2nd edition, 1994.

[10] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. The implementation of Lua 5.0. *Journal of Universal Computer Science*, 11(7):1159–1176, 2005.

[11] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. Lua: an extensible extension language. *Software: Practice & Experience*, 26(6):635–652, 1996.

[12] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

Table 2. Cable usage comparison

Cable		Human expert		No splitting		300mm		200mm		100mm		50mm		10mm	
Gauge and color	(R$/m)	(m)	(R$)	(m)	(R$)	(m)	(R$)	(m)	(R$)	(m)	(R$)	(m)	(R$)	(m)	(R$)
$0.5mm^2$ two way shielded	1.69	100.00	69.34	69.34	117.18	79.67	134.65	76.69	129.60	77.86	131.58	77.22	130.50	77.41	130.82
$0.75mm^2$ dark blue	0.26	500.00	505.71	505.71	131.48	529.41	137.65	519.62	135.10	515.61	134.06	512.56	133.26	512.37	133.22
$0.75mm^2$ black	0.26	100.00	25.81	25.81	6.71	24.71	6.42	24.33	6.33	24.30	6.32	23.99	6.24	24.33	6.33
$0.75mm^2$ red	0.26	90.00	34.83	34.83	9.06	41.98	10.92	48.13	12.51	48.22	12.54	50.94	13.24	52.39	13.62
$1.5mm^2$ yellow	0.37	30.00	13.29	13.29	4.92	14.08	5.21	11.41	4.22	11.50	4.25	11.37	4.21	11.35	4.20
$1.5mm^2$ black	0.37	100.00	87.56	87.56	32.40	86.38	31.96	87.48	32.37	88.07	32.59	85.56	31.66	81.50	30.16
$1.5mm^2$ green/yellow	0.36	40.00	43.94	43.94	15.82	43.89	15.80	43.65	15.71	43.71	15.74	43.66	15.72	43.68	15.72
$2.5mm^2$ black	0.58	30.00	13.20	13.20	7.66	13.00	7.54	13.05	7.57	13.18	7.65	12.86	7.46	12.92	7.50
$2.5mm^2$ green/yellow	0.58	30.00	16.39	16.39	9.51	16.41	9.52	16.43	9.53	15.89	9.21	15.97	9.26	15.92	9.23
$4.0mm^2$ green/yellow	0.90	20.00	1.92	1.92	1.73	1.92	1.73	1.86	1.67	1.86	1.67	1.86	1.67	1.88	1.69
Total cost			463.70		336.47		361.40		354.61		355.61		353.22		352.49