# 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 Elétricos e Mecânicos, 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** This paper deals with the cable routing problem found in electrical panels, aimed on cost minimization. A heuristic solution, using an insertion algorithm and a modified version of the Dijkstra's algorithm, is proposed and analyzed. Tests have shown that good results can be obtained from common layouts.

**Keywords.** cable routing, electrical systems design, optimization

## 1. Introduction

In large industrial facilities, like hydroelectric power plants, automobile manufacturers, and other facilities equipped with medium or large-sized industrial automation systems, there are panels and cabinets using dozens or hundreds of electrical components 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 panel and in the design quality. The right 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*. This is a real NP-complete problem from the industry, with a large scope, since many other subproblems are embedded in it.

The routing may be made by hand: 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 following to the cheaper ones. If the shortest route between two components becomes overfilled, the

[1]E-mail: aittner@netuno.com.br

[2]E-mail: claudio@joinville.udesc.br

[3]E-mail: fsasse@alumni.uwaterloo.ca

designer re-routes 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 suggests computer methods for optimization of routes, aiming to minimize the global cable cost of the system.

## 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 the Figure 1.

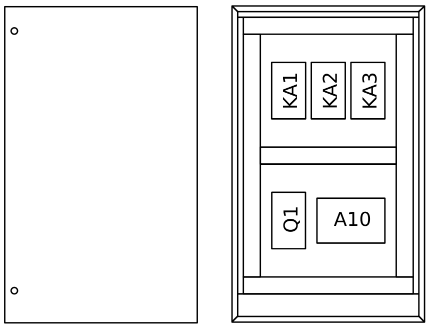

**Figure 1.** A common panel

A connection $I_n$ among components on the same panel or components of several distinct panels 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 $c_n$ 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.

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 $c_n$ 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. Permutations with inverted connection order of the terminals are not electrically distinct. Therefore, 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.

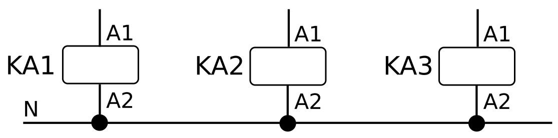

**Figure 2.** Connection among three terminals of three distinct components

A terminal is represented by

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

where $n_c$ is the component tag, $n_t$ is the identification of the connected terminal 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 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 $B_{xyz}$ denote the starting and ending points, respectively, of the conduit in the panel space and $t$ specifies the conduit type. For modeling, there are two types of conduits: (i) *open conduits*, that allows the crossing of cables through its walls, when this is admissible by design, and (ii) *closed conduits* that does not allow crossing. The Figure 3 shows conduits of both types, respectively denoted by dashed and full lines. The length of a conduit is given by the Euclidean distance between the points $A_{xyz}$ and $B_{xyz}$.

The first and the last jump of each cable may pass through the walls of a open conduit if there is no ending near 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

[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.

the cable length calculation. 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.

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 the 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.

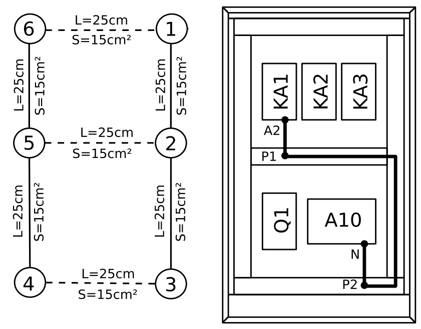

**Figure 3.** Conduit graph of the panel from Figure 1 and routed cable

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, these 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. Simplification

The model described in the 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 following simplifications may be introduced:

- Apply the routing 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.
- Use only conduits of the *closed* type. All cables must enter and leave the conduits by the endings. This simplification translates the search for entry points to a search for the nearest ending of a conduit, performed in linear time[5].

Without the complexity needed to define the cable list from the connection list, a solution to the simplified model needs only the routes attributions to each cable. 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.

The given problem has 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 complexity: the cost of a cable changes as the problem evolves since the conduits become saturated when more and more cables are inserted.

An different approach to a similar problem, using genetic algorithms was considered in [2].

## 4. The Routing Algorithm

The proposed algorithm (see Algorithm 1) routes each cable through the shortest path available in the conduit graph, decrementing the available space according to the cross-section area of the cable. This process is called *graph shrinking*.

The cost minimization comes from the pre-sorting. By inserting first the most expensive cables, the algorithm tries to avoid the saturation, giving the shortest routes to them. The cheaper cables are routed later, getting increasingly worst routes due to conduit saturation.

[5]This search have a complexity order of $O(2n_c)$ for $n_c$ conduits. There are some further feasible optimizations, not implemented in the current proposal. For example, in a assembly of several docked panels, the conduits and terminals may be grouped by panel, and the search limited to the conduits of the same panel of the terminal.

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 [3,4]. 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 $L_c$, and nodes $L_n$;
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 $L'_w$;
        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 may be found without the need of too many permutations. A solution may be considered optimal if no cable was was shifted from its shortest paths due to conduit saturation.

### *4.1. Implementation*

The Algorithm 1 was implemented in the Lua programming language[5,6]. 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.

The Dijkstra's Algorithm [7] is a classic algorithm for the single-source shortest path problem for a directed graph with non-negative edge weights. The implementation used here differs from the standard algorithm by considering only conduits with enough space for the cables. Therefore, only the adjacency matrix is needed for a single solution, lowering the need for processing. In order to allow the search on non-directed graphs, like those associated to the conduits, the adjacency matrix keeps two references for each conduit.

## 5. Tests and Results

A case-study was performed, comparing the solutions given by the implementation described above and one given by a human expert. This comparison uses the project[6] of

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

a PLC remote panel with 57 segments of conduit and 692 wires and cables connecting 1096 distinct terminals. The Figure 4 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 generated from the wiring list originally 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. All cables were routed through the panel in the first iteration of the algorithm. Table 1 shows the amount of cables calculated and the amounts bought, at time of panel assembly, with assistance of the human expert using his professional knowledge, but without any formal procedure. Costs are given in Brazilian Reais (R$, BRL in ISO 4217).

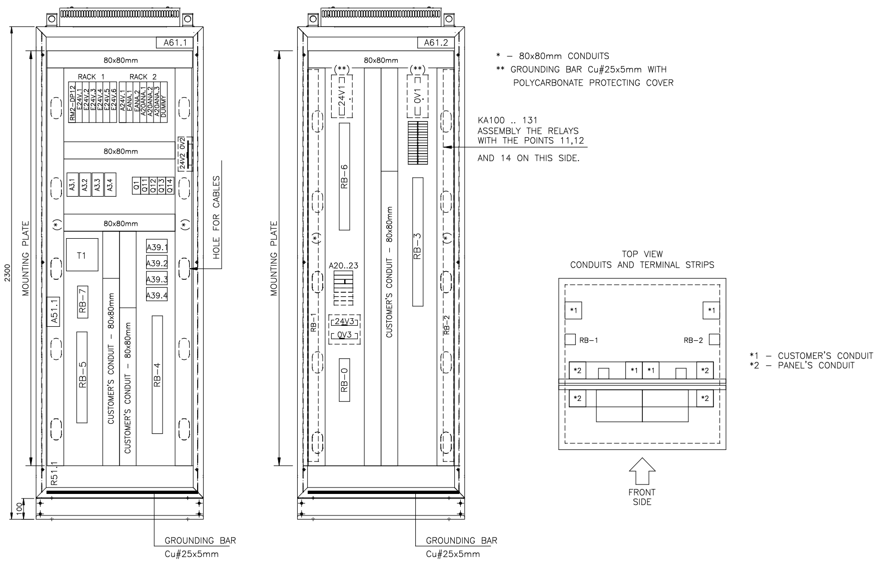

**Figure 4.** Panel from the case study (Source: WEG Automação S.A.)

The table 1 shows the properties of the route selection heuristic. It shows that in all instances of shielded cable, the most expensive one, got good routes, causing a substantial saving. 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 a extreme case). The focus of the algorithm on the cost minimization may be inferred from the global cost decrease.

This process took a mean time of 1.34s to run in a Athlon XP 2000+ computer running a Linux operating system and the standard Lua 5.1.1 interpreter. Further tests made with the LuaJIT 1.1.3 just-in-time compiler ran in a mean of 0.82s in the same machine. The running times were measured with the Unix `time` command and includes the time needed to start the interpreter and load the data files.

**Table 1.** Cable amount comparison

| Cable | | Manual method | | Heuristic method | |
|---|---|---|---|---|---|
| Gauge and color | (R$/m) | (m) | (R$) | (m) | (R$) |
| $0.5\mathrm{mm}^2$ two way shielded | 1.69 | 100.00 | 169.00 | 69.34 | 117.18 |
| $0.75\mathrm{mm}^2$ dark blue | 0.26 | 500.00 | 130.00 | 505.71 | 131.48 |
| $0.75\mathrm{mm}^2$ black | 0.26 | 100.00 | 26.00 | 25.81 | 6.71 |
| $0.75\mathrm{mm}^2$ red | 0.26 | 90.00 | 23.40 | 34.83 | 9.06 |
| $1.5\mathrm{mm}^2$ yellow | 0.37 | 30.00 | 11.10 | 13.29 | 4.92 |
| $1.5\mathrm{mm}^2$ black | 0.37 | 100.00 | 37.00 | 87.56 | 32.40 |
| $1.5\mathrm{mm}^2$ green/yellow | 0.36 | 40.00 | 14.40 | 43.94 | 15.82 |
| $2.5\mathrm{mm}^2$ black | 0.58 | 30.00 | 17.40 | 13.20 | 7.66 |
| $2.5\mathrm{mm}^2$ green/yellow | 0.58 | 30.00 | 17.40 | 13.39 | 9.51 |
| $4.0\mathrm{mm}^2$ green/yellow | 0.90 | 20.00 | 18.00 | 1.92 | 1.73 |
| **Total** | | | 463.70 | | 336.45 |

## 6. Concluding Remarks and Future Work

The proposed algorithm gave excellent results when solving the most common routing problems, when the space available in the conduits were not tightly restricted, normally getting a solution after few iterations. Problems with feasible solutions that cannot be found in the initial iterations are rare in the practical projects found in the industry. The approximation introduced by the lack of entry points did not cause considerable losses, unless for cables wiring near terminals (jumpers).

Future work will focus on methods to generate the wiring lists from the connection lists using Constraint Programming[8] and alternative ways to permute the wire list when the first iteration fails. The feasibility of a solution for the complete routing model given in the section 2 using an ant colony optimization system, similar to the system used by [9] in communication networks, is also an interesting starting point for new research.

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

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

[5] 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.

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

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

[8] I. J. Lustig and J.-F. Puget. Program does not equal program: constraint programming and its relationship to mathematical programming. *Interfaces*, 31(6):29–53, 2001.

[9] Gianni Di Caro. *Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks*. PhD thesis, Université Libre de Bruxelles, 2004.