An Improved Heuristic Solution to the Cable Routing Problem in Electrical Panels

Alexandre Erwin Ittner
Departamento de Projetos, Engenharia e Automação
WEG Automação S.A., Jaraguá do Sul, SC, Brazil
aittner@gmail.com

Claudio Cesar de Sá
Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina – UDESC, Joinville, SC, Brazil
claudio@joinville.udesc.br

Fernando Deeke Sasse
Departamento de Matemática
Universidade do Estado de Santa Catarina – UDESC, Joinville, SC, Brazil
fsasse@alumni.uwaterloo.ca

Abstract

This paper presents new results concerning the heuristic optimization of cable routes in electrical panels. The problem is modeled and a heuristic solution, using an insertion algorithm, is proposed, analyzed, and compared with previous results. Tests have shown that good results can be obtained from a common layout found in the industry.

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 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 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 in electrical panels*. This problem differs from the problem of routing cables in raceways, described in [3], due to the existence of *entry points*, as described in Section 2.

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 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 suggest an computer solution that improves the results obtained in [2] by adding the handling of open conduits and partial saturation.

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.

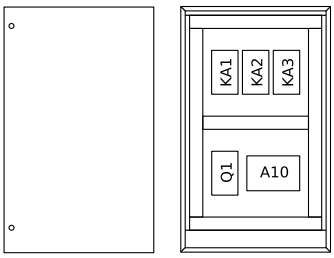

Figure 1. A typical panel

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[1]. 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.

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

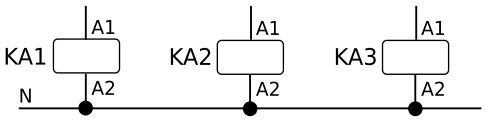

Figure 2. Connection among three terminals of three distinct components

A terminal is represented by

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

where n_c is the component, n_t is the identification of the 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} respectively denotes the starting and ending points 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. 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 closed conduit ending nearer to it. The points where the cable crosses the conduit wall are called *entry points* (points $P1$ and $P2$ in Figure 3) and makes this problem different of the problem of routing cables in electric raceways. 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.

The conduit set may be represented as a weighted graph with edges connecting nodes attributed arbitrarily, preserv-

ing 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.

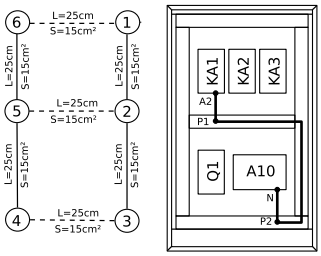

Figure 3. Graph from Fig. 1 and routed cable

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.

2.1 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.

The given problem has some similarity to the classical, NP-complete[4], *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: the cost of a cable changes as the problem evolves since the conduits become saturated when more and more cables are inserted.

2.2 Handling of Open Conduits

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 a example of a partially saturated conduit.

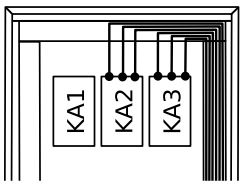

Figure 4. Partial saturation of a conduit.

This problem was 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. This process may generate too many short edges, increasing the complexity of the graph and the solving time, but without much improvements in the results. So, a new heuristic was added to limit the conduit length to a minimal: instead of splitting the conduit exactly over the entry-point, that is done at regular user-defined intervals. It also allows the user to select how much effort will be applied to the optimization of that routes.

3 The Routing Algorithm

The proposed algorithm (see Algorithm 1) runs in two steps: First, it performs the conduit splitting, using the heuristic presented in Section 2.2 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, the algorithm tries to avoid the saturation, 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 [5]. 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.

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 shifted from its shortest paths due to conduit saturation.

3.1 Implementation

The Algorithm 1 was implemented in the Lua programming language. Data files with information about the panel geometry and the wiring list are loaded using the language's own parser, ran 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 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.

4 Tests and Results

A case-study was performed comparing the solution generated by the implementation described in Section 3.1 with one solution given by a human expert. The results of the human-solved instance and the panel model used for this comparison were taken from [2]. It is the project[2] of a PLC remote panel with 57 segments of conduit and 692 wires and cables connecting 1096 distinct terminals. 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.

The test consisted 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 were 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. Since the implementation is not parallelized, it takes no advantage from multi-core CPUs. 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 using his professional knowledge, but without any formal procedure. Costs are given in Brazilian Reais.

This test shown 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 a 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.

5 Concluding Remarks and Future Work

The proposed algorithm gave excellent results when solving an instance of the problem that features the most

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

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, for that $dist(w_{start}, n_{start})$ and $dist(w_{end}, n_{end})$ are minimal;

Use Dijkstra's to find the shortest path r_w between nodes n_{start} and n_{end} of L_c, considering only the conduits with enough internal space for the w;

Update the available space in the conduits of L_c 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;

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

common routing requirements and when the space available in the conduits was not tightly restricted. In term of cost, the heuristic for handling open conduits raised better results than the ones obtained in [2] and it also allowed 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, but these times are not prohibitively high under the circumstances commonly found in the industry (e.g. panels up to 300 conduit segments).

A direct comparison between the approach proposed in this paper and the solution using Genetic Algorithms presented in [3] is limited because that problem features no entry points or open conduits. However, comparisons using panels without open conduits still possible and should be performed. The current state of the research also asks for comparisons with other optimization techniques.

Currently, the authors focus on gathering data for more case studies and new algorithms using graph-domain constraint logic programming (CLP) for generate the cable lists from the connection list, that will allow the implementation of the complete model described in Section 2. This new CLP domain was first introduced in [1] and an ECLiPSe implementation developed by [6] was used to trace paths in metabolic networks. Since the cable routing problem have some similarities with this problem, these new approaches are expected to give good results and, therefore, asking for more research.

References

[1] Y. D. G. Dooms and P. Dupont. Cp(graph): Introducing a graph computation domain in constraint programming. In *11th International Conference on Principles and Practice of Constraint Programming, LNCS, No. 3709*, pages 211–225, Sitges, Barcelona, Spain, 2005. Springer-Verlag.

[2] A. E. Ittner, C. C. de Sá, and F. D. Sasse. A heuristic approach to the cable routing problem in electrical panels. In *Proceedings of the VI Congress of Logic Applied to Technology – LAPTEC'07*, 2007.

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

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

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

[6] R. Viegas and F. Azevedo. GRASPER: A Framework for Graph CSPs. In J. Lee and P. Stuckey, editors, *Proceedings of the 6th International Workshop on Constraint Modelling and Reformulation (ModRef'07)*, Providence, Rhode Island, USA, September 2007.

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