Automated Planning to Evolve Smart Grids with Renewable Energies

Sandra Castellanos-Paez, Marie-Cecile Alvarez-Herault, Philippe Lalanda

Automated Planning to Evolve Smart Grids with Renewable Energies

Sandra Castellanos-Paez[1,2][0000−0002−6241−7974], Marie-Cecile Alvarez-Herault[2], and Philippe Lalanda[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP**, LIG, 38000 Grenoble, France
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP, G2Elab, F-38000 Grenoble, France
sandra.castellanos@univ-grenoble-alpes.fr

Abstract. Smart electrical grids play a major role in energy transition but raise important software problems. Some of them can be efficiently solved by AI techniques. In particular, the increasing use of distributed generation based on renewable energies (wind, photovoltaic, among others) leads to the issue of its integration into the distribution network. The distribution network was not originally designed to accommodate generation units but to carry electricity from the distribution network to medium and low voltage consumers. Some methods have been used to automatically build target architectures to be reached within a given time horizon (of several decades) capable of accommodating a massive insertion of distributed generation while guaranteeing some technical constraints. However, these target networks may be quite different from the existing ones and therefore a direct mutation of the network would be too costly. It is therefore necessary to define the succession of works year after year to reach the target. We addressed it by translating it to an Automated Planning problem. We defined a transformation of the distribution network knowledge into a PDDL representation. The modelled domain representation was fed to a planner to obtain the set of lines to be built and deconstructed until the target is reached. Experimental analysis, on several networks at different scales, demonstrated the applicability of the approach and the reduction in reliance on expert knowledge. The objective of further work is to mutate an initial network towards a target network while minimizing the total cost and respecting technical constraints.

Keywords: Automated Planning · Smart Grids · Distribution Network · Distributed Generation.

1 Introduction

Distribution power grids have historically been developed to deliver electricity from the transmission grid to the final customers. For instance, in France, the total length of the distribution grid is 1,377,269 km against 105,942 km for the

** Institute of Engineering Univ. Grenoble Alpes

transmission grid i.e. the slightest investment must be justified by a technical and economic calculation. In a context of unidirectional power flows with customers having a relatively well-known electrical behaviour, rules for the construction and development of the grid have been defined based on realistic assumptions of load growth.

Many changes have started to appear in recent years at varying speeds depending on the country. First of all, distributed generations (DGs) based on renewable energies have increased considerably due to various regulations aiming at increasing their penetration rate. Because of their small size, these productions are connected to the distribution grid which was not initially sized to accommodate a high amount of DGs. In addition, the decarbonization of transport is leading to the development of electric vehicles, which will form a new significant load on the distribution grid that could increase peak consumption significantly without a smart operation. These changes could cause over-voltage or over-current constraints in the distribution grids, requiring significant investments in order to strengthen the grid to make it more robust.

In order to anticipate these major changes, distribution grid operators (DSOs) aim to define the optimal long-term grid architecture (in a horizon of a few decades) called target grid, optimising a set of technical and economic performance indicators while respecting a set of constraints they have predefined. These modifications can be *minor*, by adding and/or removing lines and transformers, *intermediate* by creating new parts connected to the existing grid (expansion planning) or *major*, by changing the complete architecture (green-field planning) [8, 13]. Once the target grid has been determined, the set of intermediate grids allowing the transition from the initial grid to the target grid has to be defined.

Even if decision-making support tools for DSOs exist, such as calculation modules allowing the evaluation of different performance indicators, there is no tool allowing to automate the creation of target grids as well as the succession of intermediate grids.We believe that AI Planning can be applied to determine these intermediate grids. Indeed, AI planning is beneficial to address problems subject to continuous change, a large number of high-performance planners (implementing different approaches) are already available and planning algorithms are constantly evolving.

In this paper, we investigate how AI planning techniques can be leveraged to address the evolution of smart grids. More specifically, we address the dynamics of line connections of a distributed network. This task was previously tackled by hand and therefore with this work we enable new levels of automation. More precisely, in Section 2 we present the context of distribution systems while in Section 3 we introduce the AI planning notions used in this work. In Section 4, we show our initial investigation on how AI Planning can be used for Smart Grids. Finally, we conclude in Section 5 by discussing our results and giving some perspectives.

2 Distribution System Planning Rules

In France, the most common traditional architecture is the *secured feeder* shown in Figure 1. Each primary substation (square) is connected to another primary substation via a set of electric lines, called main feeders (lines), supplying secondary substations (small and big circles). Two connection modes are possible depending on whether the grid is in a rural or urban environment. In rural areas, the load density being low, the main feeder passes close to the secondary substations which are connected to them via secondary feeders also called antennas . In urban areas, the load density being high, the main feeder is directly connected to secondary substations. Each secondary substation is connected to the main or secondary feeder via two remote-controlled or manual switches. For the grid to be radial, all of these switches are normally closed (small circle) except one (in the big circle). The advantage of having a loopable grid is to be able to reconfigure it in case of fault. Normally *open* switches are always remotely controlled, but for normally *closed* switches this is only the case for a few (techno-economic compromise). Switches are manually or remotely operated in order to isolate the faulty portion of the feeder so that customers can be re-energised while field crew carry out repairs. In this work, we consider grids in urban environments. We also

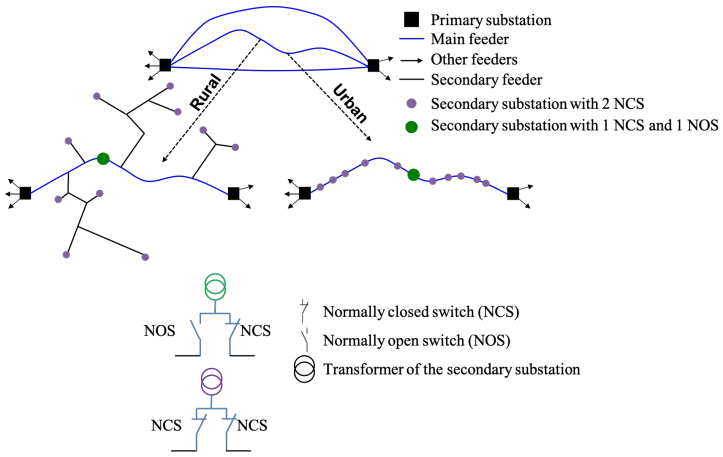


Fig. 1: Secured feeder architecture.

assume that the provided target grid, optimising a set of technical and economic performance indicators while respecting a set of constraints, is reachable from the provided initial grid. In other words, both grids have the same primary and secondary substations and differ only by their connections and their normally open switches. Additionally, we are focused in the transition from the initial grid to the target grid, i.e. the intermediate grids. Finally, in this initial investigation, we only address the dynamics of line connections.

3 Automated Planning

Automated planning (AI planning), a sub-field of Artificial Intelligence, is a model-based approach to action selection [6]. It aims to study and design domain-independent general approaches to planning[3] [7]. The dynamics of the domain of interest, their possible actions and the conditions to attain some goal are expressed on a high-level description of the world, namely *the planning model*. By using AI planning, the development of a domain-specific solver, which would cost time and money, is not necessary – it comes down to write the planning model. Thus, AI planning represents a cost-effective method for quickly setting up a solver since planning models can be more human readable and easier to modify. In the following, we present the formal definition of the planning key concepts used in this work.

3.1 Key concepts

Because the interest of planning lies in choosing actions to transform the system state, the transitions between states are represented with a state-transition system model. A state s is a set of predicates, i.e. a set of logical propositions. We address sequential planning in the STRIPS framework [5].

A planning task consists of a planning domain Σ and a planning problem $\mathcal{P}$. A classical planning domain is a restricted state-transition system $\Sigma = (S, A, \gamma)$ such that:

- S is included in the set of all states that can be described with the representation language $\mathcal{L}$.
- A is the set of all actions a.
- $\gamma(s, a)$ is the state-transition function that defines the transition from a state s to an state s' using an action a.

A classical planning problem $\mathcal{P}$ is defined over a domain Σ as $\mathcal{P} = (\Sigma, s_0, g)$ being s_0 an initial state where $s_0 \in S$ and g a goal, namely a set of instantiated predicates. A goal is satisfied if the system attains a state s_g such that all predicates in g are in s_g.

A planning operator is a triple o = *(name_o, precond_o, effects_o)* where *name_o* is in the form $name\_o(x_1, ..., x_n)$ such that $x_1, ..., x_n$ are the object variable symbols that appear in o, *precond_o* is the set of predicates that must hold before exploiting the action and *effects_o* is the set of predicates to be applied to a state.

An action a is an instantiation of a planning operator. Thus, a is a triple $a = (pre\_a, add\_a, del\_a)$. If an action can be applied, a new state is generated. First it deletes all instantiated predicates given in the delete list del_a, also known as the negative effects. Then, it adds all instantiated predicates given in the add-list add_a, also known as the positive effects.

[3] Here we refer to planning as the problem of finding a sequence of actions to achieve a goal.

A state s' is reached from s by applying an action a according to the transition function in (1).

$$s' = \gamma(s, a) = (s - del(a)) \cup add(a). \tag{1}$$

The application of a sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ to a state s is recursively defined in (2).

$$\gamma(s, \langle a_1, \ldots, a_n \rangle) = \gamma(\gamma(s, a_1), \langle a_2, \ldots, a_n \rangle). \tag{2}$$

A *satisfying* plan is an ordered sequence of actions $\pi = \langle a_1, \ldots, a_n \rangle$ such that $s_g = \gamma(s_i, \pi)$ satisfies the goal g and the latter is reachable if such a plan exists. The plan is *optimal* if it is the shortest possible path.

3.2 PDDL representation language

The planning model is written using a representation language. One of the languages used in AI Planning (and selected for this work) is called PDDL. It stands for Planning Domain Definition Language [11, 12]. It was introduced in 1998 for the International Planning Competition with the aim of standardising the planning representation language.

The planning model is then written as a compact representation of a planning task in PDDL-Code. The domain is composed of *predicates* which characterise the properties of the objects and a set of non-instantiated *actions* which establish the ways to move from one state to another. The problem is composed of *objects* which define the task relevant things in the world; an initial state s_i which represents the starting configuration of the world; and a goal state g which describes the desired predicates that we want to be true.

In Figure 2, we show a general representation of a planning task in PDDL. The domain definition (Figure 2a) is then a general model while the problem definition (Figure 2b) is a specific problem instance.

```
(define (domain <domain name>)
    <PDDL code for predicates>
    <PDDL code for first action>
    [...]
    <PDDL code for last action>
)
```

(a) Domain definition in PDDL

```
(define (problem <problem name>)
    (:domain <domain name>)
    <PDDL code for objects>
    <PDDL code for initial state>
    <PDDL code for goal specification>
)
```

(b) Problem definition in PDDL

Fig. 2: Planning task in PDDL.

3.3 Planning systems

The main purpose of writing a planning model in PDDL is to use planning systems (also known as planners) to solve challenging problems. A planner takes as input a model of the domain and a problem instance. As a result, it generates a plan as an answer to the specified input problem.

In the last 20 years, the automated planning community has developed a variety of state-of-the-art planners able to scale up to large problems thanks to the use of effective domain-independent heuristics which allow to guide the search. The algorithms embedded in planning systems search for paths in the search space through different search strategies. They also have properties such as time and memory complexity, completeness and optimality.

From the performance results in past planning competitions [15], we can highlight the planners SymBA*-2 [14], YAHSP3 [16], BFS(f) [9] and IBaCoP2 [4]. Since it is not the aim of this work to make a complete review of neither all planning systems nor all the different approaches, we leave it to the interested reader to consult the extensive literature available [3, 6, 7].

4 AI Planning for Smart Grids: Initial Investigation

Smart electrical grids play a major role in energy transition but raise important software problems. Some of them can be efficiently solved by AI techniques [2]. In particular, the increasing use of distributed generation based on renewable energies (wind, photovoltaic, among others) leads to the issue of its integration into the distribution network.

Some methods have been used to automatically build target architectures to be reached within a given time horizon (of several decades) capable of accommodating a massive insertion of distributed generation while guaranteeing some technical constraints [1]. However, these target networks may be quite different from the existing ones and therefore a direct mutation of the network would be too costly. It is therefore necessary to define the succession of works year after year to reach the target, i.e. the intermediate grids.

As previously said, the task of determining the intermediate grids that allow to reach the target grid is done by hand. Our assumption is that AI planning could fit this task very well. First, AI planning is beneficial to address problems subject to continuous change (e.g. integration of renewable energies, management of the load caused by electric vehicles, etc). Indeed, writing and modifying a planning model is less costly and time consuming than implementing a solver and adapting it with each new change. Second, planning tools are indicated when the problem is to find a solution path in a large transition system. In our case, the intermediate grids are defined as the path allowing the transition from the initial grid to the target grid. Finally, AI planning provide a wide open space of promising algorithmic possibilities.

Running Example In Figure 3, we present a distribution network of six nodes that will allow to illustrate our approach. It consists of a primary substation

(square) connected to another primary substation via main feeders (lines) supplying secondary substations (small and big circles). The nodes have been numbered from 0 to 7 for ease of reading when we refer to them.

We recall that secondary substations with one normally closed switch (NCS) and one Normally Open Switch (big circle) provide a radial grid and allow the grid to be reconfigured in case of a fault. In the initial grid (see Figure 3a) NOS are located in nodes 1 and 4, while in the target grid (see Figure 3b) they are located in nodes 3 and 4. Additionally, we consider the following open lines (dashed line):

- Initial Grid:
  - NOS in node 1 is open towards node 2.
  - NOS in node 4 is open towards node 5.
- Target Grid:
  - NOS in node 3 is open towards node 2.
  - NOS in node 3 is open towards node 5.

(a) Initial Grid (b) Target Grid

Fig. 3: Running Example of a Distribution Network.

4.1 Modelling of the Distribution Network Knowledge

Modelling Guidelines In this initial investigation, we only address the dynamics of line connections. We consider grids in urban environments. We assume that we are provided with an initial grid and a target grid. The target grid should be reachable from the provided initial grid, i.e. both grids have the same primary and secondary substations and differ only by their connections and their normally open switches. Modelling guidelines are focused in finding the transition from the initial grid to the target grid, i.e. the intermediate grids. Each one of the them must be valid in itself. This means that an intermediate grid must follow the rules below:

- A primary substation cannot be isolated, i.e. there must always be at least one secondary substation connected to it.
- A secondary substation cannot be isolated, i.e. there must always be a path that connects it to a primary substation.

- There must be at least (but not more than) one NOS on each set of lines (i.e. main feeder) that connect two primary substations together. For example, in Figure 3a the main feeder that connects primary substations 0, 7 with secondary substations 1, 2 has only one NOS on node 1.
- A primary substation can be directly connected to one or more secondary substations. For example, the primary substation at node 0 could have another secondary substation connected to it.
- A secondary substation can be directly connected to at most two other, primary or secondary, substations. For example, the secondary substation at node 6 (see Figure 3a) already has two connections (with nodes 5 and 7) and therefore cannot have more. This constraint has been arbitrarily chosen and can be modified by slightly changing the model.

Finally, following the intermediate grids, one by one, from the initial grid must result in the target grid.

Formalisation into PDDL We address the task of determining the intermediate grids by translating it to an Automated Planning task. The real challenge is to write the domain model. We therefore need to define the main elements of a PDDL planning task (see Section 3.2) in the distribution network domain. These elements are:

- Predicates:
  - Is there an open line between x substation and y substation? (PDDL-Code 1.1 line 5)
  - Is x substation connected to y substation? (line 6)
  - Is x substation feed by p primary substation? (line 7)
  - Is s secondary substation free to receive another connection? (line 8)
  - Is x substation mutable? (line 9)
- Actions/Operators:
  - A line between two secondary substations can be removed if there exists a normally open switch between them. For example, we can close the NOS at 4 and remove the line between nodes 4 and 5. (PDDL-Code 1.1 line 11)
  - A line between a secondary substation and a primary substation can be removed if there exists a normally open switch between them. (line 23)
  - A secondary substation can change the direction of its normally open switch. For example, the NOS of the secondary substation at node 4 is open towards node 5 but it can be closed here and be open for the line between 3 and 4. (line 34)
  - Two secondary substations can be connected and a NOS can be placed in one of them towards the other substation, if each of them can still receive an additional connection. (line 47)
  - A secondary substation can be connected to a primary substation and a NOS can be placed in the secondary substation the primary substation, if the secondary substation can still receive an additional connection. (line 60)

Notice that each action has a set of precondition which determines if they can be applied in a given state. If the action can be applied, it alters the set of true facts according to the action's effects. We provide in PDDL-Code 1.1, the complete definition of predicates and actions.

```
(define (domain DISNET)
  (:requirements :typing)
  (:types primary secondary - substation)
  (:predicates
    (open_line ?x - substation ?y - substation)
    (connected ?x - substation ?y - substation)
    (feed ?x - substation ?p - primary)
    (free_for_connection ?s - secondary)
    (mutable ?x - substation)
  )
  (:action remove_openLine_secTosec
    :parameters (?s1 - secondary ?s2 - secondary)
    :precondition (and
        (open_line ?s1 ?s2) (connected ?s1 ?s2)
        (mutable ?s1) (mutable ?s2))
    :effect (and (not(connected ?s1 ?s2))
        (not(connected ?s2 ?s1))
        (not(open_line ?s1 ?s2))
        (not(open_line ?s2 ?s1))
        (free_for_connection ?s1)
        (free_for_connection ?s2)
        ))
  (:action remove_openLine_secTopri
    :parameters (?s1 - secondary ?p1 - primary)
    :precondition (and
        (open_line ?s1 ?p1) (connected ?s1 ?p1)
        (mutable ?s1))
    :effect (and (not(connected ?s1 ?p1))
        (not(connected ?p1 ?s1))
        (not(open_line ?s1 ?p1))
        (not(open_line ?p1 ?s1))
        (free_for_connection ?s1)
        ))
  (:action change_switch_connection
    :parameters (?s1 - secondary ?s2 - substation
    ?s3 - substation ?p1 - primary ?p2 - primary)
    :precondition (and
        (open_line ?s1 ?s2) (connected ?s1 ?s2)
        (connected ?s3 ?s1) (feed ?s3 ?p1)
        (feed ?s1 ?p1) (feed ?s2 ?p2))
    :effect (and (not(open_line ?s1 ?s2))
        (not(open_line ?s2 ?s1))
        (not(feed ?s1 ?p1))
        (open_line ?s1 ?s3)
        (open_line ?s3 ?s1)
```

```
        (feed ?s1 ?p2)))
  (:action open_line-connect_secTosec
    :parameters (?s1 - secondary ?s2 - secondary)
    :precondition (and
        (free_for_connection ?s2) (mutable ?s2)
        (free_for_connection ?s1) (mutable ?s1))
    :effect (and
        (not(free_for_connection ?s2))
        (not(free_for_connection ?s1))
        (open_line ?s1 ?s2)
        (open_line ?s2 ?s1)
        (connected ?s2 ?s1)
        (connected ?s1 ?s2)
        ))
  (:action open_line-connect_secTopri
    :parameters (?s1 - secondary ?p1 - primary)
    :precondition (and
        (free_for_connection ?s1) (mutable ?s1))
    :effect (and
        (not(free_for_connection ?s1))
        (open_line ?s1 ?p1)
        (open_line ?p1 ?s1)
        (connected ?p1 ?s1)
        (connected ?s1 ?p1)
        )))
```

PDDL-Code 1.1: Definition of the distribution network domain.

As previously said, writing the domain model is challenging. Additionally, it is also necessary to verify that the domain is expressive enough to specify a problem. We therefore need to write the problem specification. To illustrate this, we have the following elements for the running example:

- Objects:
  - Two primary substations. $P1$ refers to node 0 and $P2$ refers to node 7.
  - Six secondary substations. S_i refers to node i where $i \in 1, 2, 3, 4, 5, 6$.
- Initial state:
  - A *connected(x,y)* predicate and its mirror, for each two connected substations x and y in the initial grid.
  - A *open_line(x,y)* predicate and its mirror, for each open line between two substations x and y in the initial grid.
  - A *feed(x,p)* predicate, for each substation x feed by a primary substation p in the initial grid.
  - A *mutable(x)* predicate, for each substation x in the initial grid that changes in comparison to the target grid. For example, the secondary substation at node 1 is *mutable* since in the initial grid it is connected to node 2 but in the target grid it is not. On the contrary, the secondary substation at node 6 is *not mutable* since in the initial and in the target grid it is connected to the same nodes (i.e nodes 5 and 7).

– Goal specification:
  - A *connected(x,y)* predicate and its mirror, for each two connected substations x and y in the target grid.
  - A *open_line(x,y)* predicate, for each open line between two substations x and y in the target grid.
  - A *feed(x,p)* predicate, for each substation x feed by a primary substation p in the target grid.

We provide in PDDL-Code 1.2, the complete definition of the initial state and the goal specification for the running example in Figure 3.

```
(define (problem disnet001) (:domain disnet)
(:objects
  P1 P2 - Primary
  S1 S2 S3 S4 S5 S6 - Secondary)
(:init (mutable S1) (mutable S2) (mutable S3) (mutable S4)
  (mutable S5) (connected P1 S1) (connected S1 P1)
  (connected S1 S2) (connected S2 S1) (connected P2 S2)
  (connected S2 P2) (connected P1 S3) (connected S3 P1)
  (connected S3 S4) (connected S4 S3) (connected S4 S5)
  (connected S5 S4) (connected S5 S6) (connected S6 S5)
  (connected P2 S6) (connected S6 P2) (feed P1 P1)
  (feed S1 P1) (feed S3 P1) (feed S4 P1) (feed P2 P2)
  (feed S2 P2) (feed S6 P2) (feed S5 P2) (open_line S1 S2)
  (open_line S2 S1) (open_line S4 S5) (open_line S5 S4))
(:goal (and (connected P1 S1) (connected S1 P1)
  (connected S1 S4) (connected S4 S1) (connected S4 S2)
  (connected S2 S4) (connected S3 P1) (connected P1 S3)
  (connected S3 S5) (connected S5 S3) (connected S6 S5)
  (connected S5 S6) (connected P2 S2) (connected S2 P2)
  (connected P2 S6) (connected S6 P2) (open_line S4 S2)
  (open_line S2 S4) (open_line S3 S5) (open_line S5 S3)
  (feed P1 P1) (feed S1 P1) (feed S4 P1) (feed S3 P1)
  (feed P2 P2) (feed S2 P2) (feed S6 P2) (feed S5 P2))))
```

PDDL-Code 1.2: Definition of a distribution network problem.

4.2 Experimental Results

The modelled domain and each problem instance were fed to the DFS+ [10] planner. The planner was run with a timeout of 10 seconds and 2GB of RAM.

The planner has always successfully found a plan for each of our problem instances. In Table 1, we present for each tested problem instance the parameters of the initial and the final grid (number of primary and secondary substations, the number of main feeders), the number of steps in the obtained solution and the time, in seconds, taken by the AI planner to obtain it.

Obtained plans describe the set of lines to be built and deconstructed from the initial grid until the target grid is reached. In PDDL-code 1.3, we show the

Table 1: Description of the problem instances and obtained results.

Problem instance	Primary Substations	Secondary Substations	NOS Initial Grid	NOS Target Grid	Plan Length	Search Time (s)
disnet001	2	6	2	2	7	0.002
disnet002	2	6	2	1	13	0.003
disnet003*	2	10	2	2	35	0.07
disnet004	2	10	2	3	23	0.021
disnet005	2	20	4	4	41	1.75

seven step solution found by the planner for the problem instance disnet001 (i.e. our running example). In Figure 4, we provide a graphical representation of this plan, i.e. the intermediary grids and actions from the initial grid until reach the target grid.

```
(remove_openLine_sectosec s5 s4)
(remove_openLine_sectosec s2 s1)
(open_line-connect_sectosec s1 s4)
(change_switch_connection s4 s1 s3 p1 p1)
(remove_openLine_sectosec s4 s3)
(open_line-connect_sectosec s3 s5)
(open_line-connect_sectosec s2 s4)
```

PDDL-Code 1.3: Plan for the problem instance of the running example (disnet001).

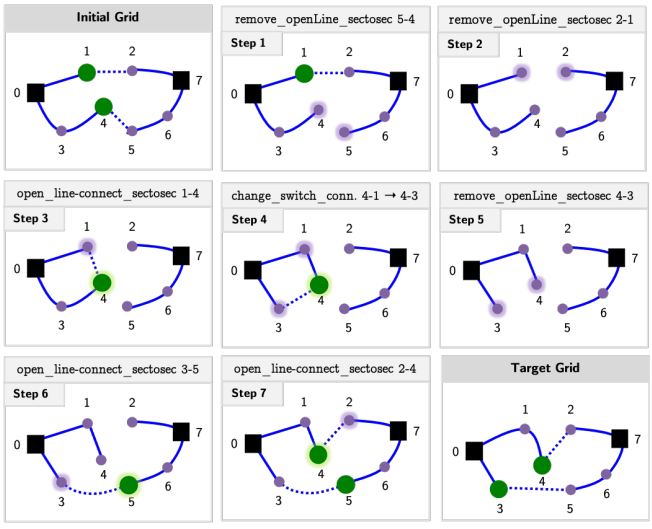
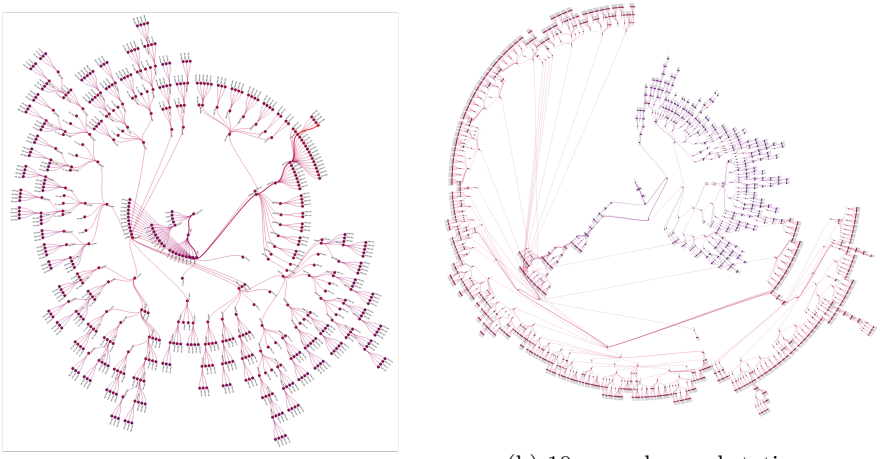


Fig. 4: Graphical representation of the plan in PDDL-Code 1.3.

5 Discussion and Conclusion

Our results clearly show that AI planning is a very promising way to solve smart grid engineering problems. The AI planner, provided with our planning model, was able to solve all problem instances very quickly (less than two seconds). In this regard, we asked a smart grid expert to solve the problem instance `disnet003` without the help of AI planning. This problem instance consist of two primary substations, ten secondary substations and two feeders (see Table 1). The solution given by the expert is 21 actions long and was found in about one hour (3600 seconds). In contrast, the solution given by the AI planner is 35 actions long and was found after 0.07 seconds.

Given the time taken by the expert to solve an easy instance, we can intuitively think that increasing the number of substations could become problematic. Indeed, if we only consider from the space of all possible grids, those where all secondary substations are part of a single main feeder then we can give a lower bound on the number of possible grid configurations. If s is the number of secondary substations then we have a lower bound in $O(s!)$ which grows over-exponentially. To illustrate this, in Figure 5, we show a representation of the explored state space for the instance `disnet001` with six secondary substations in comparison with the instance `disnet003` with ten secondary substations. We go from 568 visited states (Figure 5a) to 2795 (Figure 5b), and from a height of the search tree of 11 to 35. Because of this fast growing complexity, this task can quickly become intractable for human experts.



(a) 6 secondary substations (disnet001).

(b) 10 secondary substations (disnet003*).

Fig. 5: Representation of the explored state space for two different problem instances.

Regarding the plan length, our solution (35 actions) is longer than the one given by the expert (21 actions). This could be a disadvantage for AI planning. However, some of the actions given by the AI planner can be easily simplified to a single action and, therefore, our plan can be shortened. In particular, this is the case for the action `change_switch_connection` since the planner acts locally to ensure that each intermediate grid is valid. For example, let us imagine that we have the grid 0-1=:2-3-4-5 where '-' denotes a connection, '=' denotes an open line and ':' denotes a NOS. If the solution is 0-1-2-3-4:=5, the AI planner has to go through each intermediate change, i.e. 0-1=:2-3-4-5, 0-1-2:=3-4-5, 0-1-2-3:=4-5 and 0-1-2-3-4:=5. This can be solved in a plan post-processing step with a simple script. By applying such a post-processing step, we obtained a shortened plan of 25 actions. In addition, the plan could be further optimised to avoid unnecessary actions.

Finally, AI planning has already demonstrated its applicability to build a system for defining substation voltage targets for the Grendon substation, near London, England [2]. Authors proposed a numeric, non-temporal planning model. Our results and that work only encourage us to continue exploring AI planning for smart grids.

The objective of further work is therefore to mutate an initial network towards a target network while minimising the total cost and respecting technical constraints.

References

1. Alvarez, M.C., Caire, R., Raison, B., Enacheanu, B., Devaux, O., Jeannot, R., Hadjsaid, N.: Distribution network long term planning methods comparison with respect to dg penetration. In: I-SUP 2008 (Innovation for Sustainable Production) (2008)
2. Bell, K., Coles, A., Fox, M., Long, D., Smith, A.: The application of planning to power substation voltage control. In: ICAPS Workshop on Scheduling and Planning Applications (SPARK). pp. 1–8 (2008)
3. Castellanos-Paez, S.: Learning routines for sequential decision-making. Ph.D. thesis, Université Grenoble Alpes (2019)
4. Cenamor, I., De La Rosa, T., Fernández, F., et al.: Ibacop and ibacop2 planner. IPC 2014 planner abstracts pp. 35–38 (2014)
5. Fikes, R., Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence **3-4**(2), 189–208 (1971)
6. Geffner, H., Bonet, B.: A concise introduction to models and methods for automated planning. Synthesis Lectures on Artificial Intelligence and Machine Learning **8**(1), 1–141 (2013)
7. Ghallab, M., Nau, D., Traverso, P.: Automated planning: theory and practice (2004)
8. Khator, S.K., Leung, L.C.: Power distribution planning: A review of models and issues. IEEE Transactions on Power Systems **12**(3), 1151–1159 (1997)
9. Lipovetzky, N., Geffner, H.: Width and serialization of classical planning problems. In: ECAI 2012, pp. 540–545. IOS Press (2012)
10. Lipovetzky, N., Geffner, H.: Width-based algorithms for classical planning: New results. In: ECAI 2014, pp. 1059–1060. IOS Press (2014)

11. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: Pddl-the planning domain definition language (1998)
12. McDermott, D.M.: The 1998 ai planning systems competition. AI magazine **21**(2), 35 (2000)
13. Temraz, H.K., Quintana, V.H.: Distribution system expansion planning models: an overview. Electric Power Systems Research **26**(1), 61–70 (1993)
14. Torralba, A., Alcázar, V., Borrajo, D., Kissmann, P., Edelkamp, S.: Symba*: A symbolic bidirectional a* planner. In: International Planning Competition. pp. 105–108 (2014)
15. Vallati, M., Chrpa, L., Mccluskey, T.L.: What you always wanted to know about the deterministic part of the international planning competition (ipc) 2014 (but were too afraid to ask). The Knowledge Engineering Review **33** (2018)
16. Vidal, V.: Yahsp3 and yahsp3-mt in the 8th international planning competition. Proceedings of the 8th International Planning Competition (IPC-2014) pp. 64–65 (2014)