The Power Supply Restoration Benchmark for ICKEPS-07

Sylvie Thiébaux
National ICT Australia &
Computer Sciences Laboratory
The Australian National University
Canberra, ACT 0200
Sylvie.Thiebaux@anu.edu.au

Abstract

This document provides the material necessary to deal with the power supply restoration (PSR) benchmark (Thiébaux & Cordier 2001), as featured in the International Competition on Knowledge engineering for Planning & Scheduling (ICKEPS-07).

Introduction

The problem of restoring supply in a faulty power distribution system is of major concern for electricity distributors. It consists in localising the faulty lines on the distribution network and reconfiguring the network so as to isolate these lines and resupply most consumers affected by the faults. This has to be done within minutes. When reconfiguring, a few parameters such as breakdown costs should ideally be optimised, without violating capacity constraints and overloading parts of the network.

While partial observability is one of the most challenging features of the full PSR benchmark (Thiébaux & Cordier 2001), we will assume full observability for the purpose of the competition. That is, we will know the location of the faults, the powers consumed on the lines, and the current network configuration. A fully observable version of PSR featured in the 2004 International Planning Competition (IPC) (Hoffmann *et al.* 2006). However, that version did not take into account capacity constraints and power optimisation criteria, leading to a supply restoration problem solvable optimally in polynomial time (Helmert 2006). In contrast, (the second and third difficulty levels of) the ICKEPS competition version deals with those aspects.

The material is organised as follows. We first provide a natural language description of the benchmark. We then describe the PSR simulator used in the competition, followed by the problem examples provided to the participants. Finally, we mention other resources and papers available on PSR that can be of help.

Natural Language Description

This section borrows from (Thiébaux & Cordier 2001) in describing the restricted version of the power supply restoration benchmark used in the competition. We start by a description of the physical characteristics of power distribution systems, and follow with an explanation of the supply restoration problem.

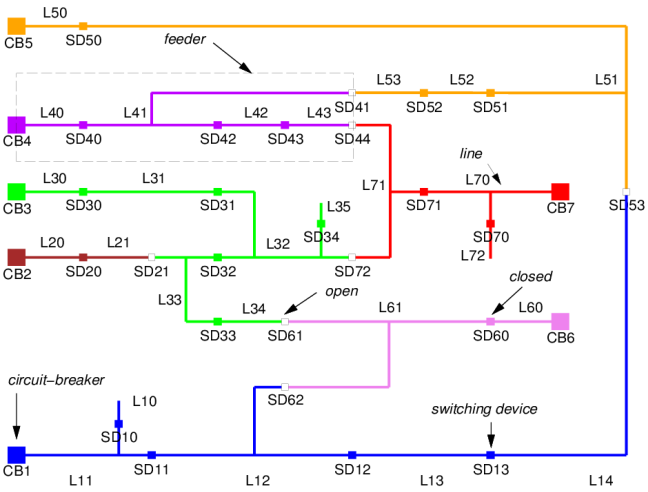

Figure 1: Power Distribution System (part)

Power Distribution Systems

Topology A power distribution system, as in Figure 1, can be viewed as a network of electric lines connected via switching devices (SDs), represented by small squares in the figure, and fed via circuit-breakers (CBs), represented by large squares. Lines can be connected to an arbitrary number of devices. Switching devices however are connected to at most two lines and circuit-breakers to at most one. Switching devices and circuit-breakers (which we collectively refer to as "devices") have two possible positions: either open or closed. White devices in the figure are open, see e.g. SD61, and the others are closed, see e.g. SD60. A circuit-breaker supplies power iff it is closed, and a switching device stops the power propagation iff it is open. We say that a device (resp. a line) is fed by a circuit-breaker when there is a path from this circuit-breaker to the device (resp. line) consisting entirely of closed devices. Consumers may be located on any line, and are then only supplied when their line is fed. Lines are marked as critical when they supply critical customers such as hospitals.

The distribution networks we are dealing with have a meshable structure exploited radially: the positions of the devices are set so that the paths taken by the power of each circuit-breaker form a tree called a feeder. The root of a feeder is a circuit-breaker, and its leaves are whatever switching devices downstream happen to be open at the time. In most cases, each line belongs to one feeder at a time. For illustration, the boxed area in the figure shows one of the feeders, and adjacent feeders are distinguished using different colors. In certain circumstances, it is possible for a line to be fed by multiple circuit-breakers, i.e., to belong to more than one feeder. In that case, these circuit-breakers are leaves of each other's feeder. We will always assume that there is no fed loop in the network, and it is an error to create such loop. For our example in Figure 1, this implies that the switching devices that are open on the figure cannot all be closed.

Size Many European power distribution systems are composed of some hundreds of feeders (typically from 100 to 300), each of which contains a few remote-controlled switching devices (the objective is to equip each feeder with 2-3 of them). A feeder has only a very few neighbours (typically from 1 to 4), and as will be seen below, essentially only those will play a role in supply restoration. Hence reasoning is very local, and the network in Figure 1 is a good representative of the complexity of the real problem. The problem can trivially be scaled up or down by modifying the number of switching devices per feeder and the number of neighbours of feeders.

Faults Power distribution systems are often subject to permanent[1] faults (short circuits) occurring on one or even several lines. Since these short circuits are mainly due to bad weather conditions and lightning, multiple faults are not rare. Upon occurrence of a fault, the circuit-breaker feeding the faulty line opens in order to protect the rest of its feeder from damaging overloads. For instance, if a fault occurs on line L13, CB1 will open. As a result, all consumers located on that feeder (lines L10-L14) are left without power. Simply reclosing the circuit-breaker will not help. Since the fault is permanent, the circuit-breaker will still be feeding it and will open again. Instead, the faulty lines must be located (a problem we assumed solved here) and the network reconfigured so as to isolate them and restore the supply to the non-faulty lines. This has to be done within a few minutes.

Actions The switches and circuit-breakers are remote controlled. We can therefore remotely open switching devices so as to isolate suspected lines and close others to direct the power from other feeders towards the non-faulty lines. In fact, opening and closing devices are the only available actions in our problem. For instance, following a fault on L13 and the resulting opening of CB1, opening devices SD12 and SD13 will isolate line L13, while subsequently closing SD53 will direct the power from CB5 in such a way as to re-supply L14. Failing to open SD13 before closing SD53 would lead CB5 to feed the faulty line L13 and therefore to open just as CB1 did.

Powers At any time, circuit-breakers and lines can only support a certain maximal power. For the purpose of checking these power constraints, the following is given:

- $pmax(c)$: the capacity of circuit-breaker c,
- $pmax(l)$: the capacity of line l
- $pcon(l)$: the power consumed by the customers on line l.

The constraints to be checked are:

1. $|pent(c)| < pmax(c)$ for every circuit-breaker c
2. $pent(l) < pmax(l)$ for every line l

where $pent$, the power entering a line or a circuit-breaker, is defined from $pcon$ below.

Assuming that most participants will not be keen to open their high-school electricity book, let us describe what needs to be done in pure algorithmic terms. We start with a few notations. Recall that each switching device is connected to at most two lines, and so has at most two "sides" – let's arbitrarily pick one of each device's sides as "up" and the other as "down". $\bar{\alpha}$ denotes the side opposite to side α. Let $succ(d, \alpha)$ be the line directly connected to side α of device d (this is undefined if no such line exists), and let $succ(l)$ be the set of pairs (device, side) directly connected to line l. For instance, naming "up" the left-hand side and "down" the right hand side of each device in Figure 1, we have $succ$(SD11,down) = L12, and $succ$(L12) = {(SD62,up), (SD12,up), (SD11,down)}.

Now, consider the feeder fed by some circuit-breaker c and any device d on this feeder. Observe that the power supplied by c enters d via exactly one of its sides, which we call $side(c, d)$. For instance, $side$(CB1,SD12) = up. If d is not on c's feeder, then $side(c, d)$ is undefined. For a circuit-breaker c, we take the convention that $side(c, c) = \bar{\alpha}$ where α is the side connected to the network, so in our example $side$(CB1,CB1) = up.

We first define the power $pent_c(d, \alpha)$ from circuit-breaker c entering side α of device d as follows, taking the convention that it is positive when entering via the up side of a device and negative when entering via the down side:

- when d is closed, $side(c, d) = \text{up} = \alpha$, and $succ(d, \bar{\alpha})$ is defined, then:

  $$pent_c(d, \alpha) = pcon(l') + \sum_{(d',\alpha') \in succ(l'), d' \neq d} |pent_c(d', \alpha')|$$

  where $l' = succ(d, \bar{\alpha})$
- when d is closed, $side(c, d) = \text{down} = \alpha$, and $succ(d, \bar{\alpha})$ is defined, then:

  $$pent_c(d, \alpha) = -pcon(l') - \sum_{(d',\alpha') \in succ(l'), d' \neq d} |pent_c(d', \alpha')|$$

  where $l' = succ(d, \bar{\alpha})$
- otherwise $pent_c(d, \alpha) = 0$.

[1] Technically, a permanent fault is one that cannot be eliminated by automatic protection devices such as shunts and reclosers.

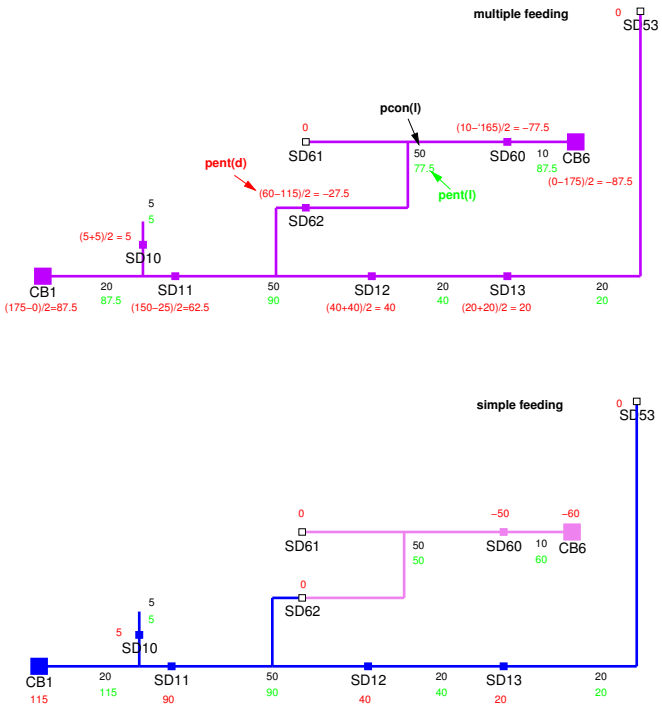

Figure 2: Power Propagation Examples. Top: multiple feeding of the same feeder; Bottom: single feeding of each feeder. $pcon(l)$ are in black, $pent(d)$ are in red, $pent(l)$ are in green. Left and bottom sides of devices are taken to be "up", and right and top sides of devices are taken to be "down".

Now, the more general formula below defines the power entering a device by dropping the references to a particular circuit-breaker and side and taking into account possible feeding of the same feeder by multiple circuit-breakers by assuming equal repartition of the load among those circuit-breakers:

$$pent(d) = \sum_{\alpha \in \{\text{up,down}\}} \frac{\sum_{c \in \text{breakers s.t. } c \text{ feeds } d} pent_c(d, \alpha)}{||\{c \in \text{breakers s.t. } c \text{ feeds } d\}||}$$

It yields a positive or negative value depending on how much power enters each side of d.

Finally, the power entering a line comes from devices whose down side is connected to the line and whose $pent$ power is positive, and from devices whose up side is connected to the line and whose $pent$ power is negative:

$$pent(l) = \sum_{(d,\text{down}) \text{ connected to } l \text{ s.t. } pent(d)>0} pent(d, \text{down}) - \sum_{(d,\text{up}) \text{ connected to } l \text{ s.t. } pent(d)<0} pent(d, \text{up})$$

Figure 2 gives two examples of calculation of $pent(d)$ and $pent(l)$ from $pcon(l)$ for a subnet of Figure 1: one in which the two feeders are as in Figure 1, and one in which they are merged into a single multiply fed feeder by closing SD62.

Supply restoration

Plans The problem of supply restoration is that of reconfiguring the network in order to resupply the consumers following the loss of one or more feeders. Under full observability, it amounts to building a sequential (or parallel) plan consisting of opening/closing operations. This plan should isolate the faulty lines by prescribing to open the switching devices surrounding them. It should also restore supply to the non-faulty areas of the lost feeders by prescribing to operate devices so as to direct the power towards these areas.

For instance, assume there is a fault on L13 and one on L10. This causes CB1 to open. A possible restoration plan is to open SD12, SD13, and SD10 to isolate the faults, to close CB1 to resupply L11 and L12 and close SD53 to have CB5 resupply L14. Alternatively, we could use CB6 instead of CB1 to resupply L11 and L12.

Constraints As we mentioned before, plans that (even temporarily) create fed loops in the network are invalid. Valid plans must comply with the power capacity constraints defined previously. This might prevent directing the power through certain paths and resupplying all the non-faulty areas. For instance, it might be the case that CB5's capacity or that of a line on its feeder be exceeded if we attempt to resupply L14.

Supply restoration systems often restrict the search space of possible plans to so called level-1 plans: plans which only *extend* existing feeders. That is, the plan does not transfer any of the load that a healthy feeder had at the time of the incident to another feeder. For instance, a plan that closes SD41 and open SD51 to discharge CB5 so that it could resupply L14 is not a level-1 plan. Other types of plans are rarely used in practice because they require a complex protocol with the dispatching center. It is up to the competitors to decide whether they prefer generating level-1 plans or not.[2]

Optimisation Criteria A good plan will optimize certain parameters under the constraints. Naturally, breakdown costs should be minimised, i.e., as much load as possible should be resupplied as fast as possible, with priority being given to critical consumers. Ideally the number of switching operations should also be optimised, so as to stay close to the configuration in which the network is normally exploited (called the normal configuration), and power margins of circuit-breakers should be balanced in anticipation of the next load peak.

We consider two alternative models for the cost of a plan: one in which the cost only depends on the final state of the plan, and the other in which intermediate states also have costs, making PSR a true sequential optimisation problem. We start by defining the various criteria relevant to the cost of state, and then use them to define the cost of a plan under the two models.

State cost criteria. For a given network state s, we define the following criteria:

- $critical(s) = ||\{l \text{ is a critical line and l is not fed}\}||$ is

[2]Level-1 as we use it here is not related to the difficulty level 1 of the competition introduced later.

the number of lines not supplied in s,

- $margin(s) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(pmax(c_i) - |pent(c_i)| - \frac{1}{n}\sum_{j=1}^{n}(pmax(c_j) - |pent(c_j)|))^2}$, is the standard deviation of the power margins of all n circuit-breakers in s,
- $breakdown(s) = \sum_{l \in \text{lines}} pcons(l) - \sum_{c \in \text{breakers}} |pent(c)|$, or equivalently $breakdown(s) = \sum_{l \in \text{ lines, } l \text{ is not fed}} pcon(l)$, is the load that is not supplied in s.

Simple plan cost model. Under this model, the cost of a plan depends on the final state s_n reached at the end of the plan execution and of the number of steps in the plan. It is defined as follows: $\beta^{i_s} step + \beta^{i_c} critical(s_n) + \beta^{i_m} margin(s_n) + \beta^{i_b} breakdown(s_n)$ where:

- β, i_s, i_c, i_m, and i_b are given integers which depend on the problem; the value of i_s, i_c, i_b, and i_m defines the priority given to each criterion, and we usually have $i_s < i_m < i_b < i_c$,
- $step$ is the number of steps of the plan,
- $critical(s)$, $margin(s)$, and $breakdown(s)$ are defined as previously.

Sequential cost model. Under this model, the cost of a plan depends on the cumulative number of critical lines and on the cumulative load not supplied during the execution of the plan, and on the circuit-breakers margins in the final state of the plan. There is no need to explicitly account for the number of plan steps since longer plans are penalised via an increase in cumulative breakdown costs. Let $s_0, s_1 \ldots s_n$ be the sequence of states reached during the execution of the plan $[o_1, o_2, \ldots, o_n]$, the plan cost is

$$\beta^{i_m} margin(s_n) + \sum_{k=0}^{n} \beta^{i_c} critical(s_k) + \beta^{i_b} breakdown(s_k)$$

where:

- β, i_c, i_m, and i_b are given integers which depend on the problem; the value of i_c, i_b, and i_m defines the priority given to each criterion, and we usually have $i_m < i_b < i_c$,
- $critical(s)$, $margin(s)$, and $breakdown(s)$ are defined as previously.

Difficulty Levels

In the competition, there are three levels of difficulty for PSR:

1. **No numerical powers.** At this level, the problem is essentially identical to that of the IPC-4 competition setting: given a faulty power system, the goal is to resupply all lines that can be, if possible in the minimum number of steps. Any plan creating a loop is treated as invalid. To be able to easily rank plans produced by participants, we give them a cost of:

   nb lines not fed in the final plan state $\times$
   nb devices in the network +
   nb plan steps

   A plan optimising that cost feeds all lines that can be fed, and does so in the minimal number of steps possible.

   Provided a non-trivial analysis, this problem can be solved in polynomial time (Helmert 2006). There is already plenty of scope to illustrate the benefits of a knowledge engineering tool at this level, for instance in facilitating the modelling of the effects of actions (which is non-trivial, see (Hoffmann *et al.* 2006; Thiébaux, Hoffmann, & Nebel 2005)) and in discovering and exploiting the structure that makes the problem polynomial.

2. **Power constraints and optimisation using the simple cost model.** At this level, numerical powers, capacity constraints and all the optimisation criteria are taken into account to compute the cost of a plan following the simple cost model defined earlier. Any plan violating the constraints (or creating a loop) is treated as invalid. Here the additional challenge is the modelling of the numerical power propagation, of the constraints and optimisation criteria, and the exploitation of the model for constrained optimisation. Note that maximising the number of lines (or merely critical lines) resupplied under the capacity constraints is already NP-hard, regardless of whether a shortest plan is sought (Helmert, personal communication).

3. **Power constraints and optimisation using the sequential cost model.** This is as in level 2, but the sequential cost model is used in place of the simple one. It could be argued that, unless intermediate states have costs, the fully observable version of PSR is more a constrained optimisation problem than a planning problem. This is due to the facts that, under those assumptions, there is a trivial ordering of actions that guarantee validity (namely perform all openings first) and that the order in which actions are executed does not affect the cost. This third difficulty level remedies this, making PSR a true sequential optimisation problem.

Simulation

This section is the manual of the simulator used in the competition. The simulator is implemented in Standard ML (SML). It takes as input a file describing the network and problem and another file describing the plan, and produces a simulation report. We successively describe the syntax of the input files and the content of the simulation report produced. In both files, the delimiters of comments are `(*` and `*)`.

Problem File

The problem description consists in describing the circuit-breakers, switches, lines, and normal configuration of the network, then the faults and the difficulty level. We examine each of those in turn.

Circuit-breakers A circuit-breaker is created via the `circuit_breaker` function, whose parameters are the name of the circuit breaker (a string), an initial position (`Open` or `Closed`), and a power capacity (a real). For instance,

```
val CB1 = circuit_breaker "CB1" Closed 195.0;
```

creates circuit breaker `CB1` whose name is `"CB1"`, which is initially `Closed`, and whose capacity is `195.0`. Note: the `.0` at the end of reals is compulsory.

Switches A switch is created via the `switch` function which takes as parameters the switch's name and its initial position. For instance,

```
val SD62 = switch "SD62" Open;
```

creates the switch SD14 whose name is `"SD62"` and which is initially `Open`.

Lines Recall that we identify a "line" with the smallest area of the network delimited by devices or earth. So a line with more than 2 delimiters (e.g. `L12` in Figure 1) will necessarily "branch". Recall also that each device has two sides, which we call `Up` and `Down`, respectively. Except for circuit-breakers (see below), it is indifferent which side of a device is called `Up` and which is called `Down`, as long as one calls it the same throughout the description.

A line is created via the `line` function, which takes as parameters the name of the line, the list of the pairs (device, side) to which it is connected, the capacity of this line (a real), the power subscribed by the customers on this line (another real), and a boolean indicating the presence of critical customers. For instance,

```
val L12 = line "L12" [(SD11,Down),(SD12,Up),
(SD62,Up)] 180.0  50.0 true;
```

creates the line `L12`, whose name is `"L12"`, connecting the `Down` side of `SD11`, the `Up` side of `SD12`, and the Up side of `SD62`, whose capacity is `180.0`, on which the current load is `50.0`, and which feeds critical customers. Note that it is an error if one of the switching devices mentioned in the connection list has not been previously defined, and in that case the simulator aborts with the message `Syntax or semantic error in file <problem-file>`.

If the connection list is a singleton, then one of the extremities of the line is taken to be connected to earth. For instance

```
val L10 = line "L10" [(SD10,Down)] 180.0
5.0 false;
```

defines a line connecting the `Down` side of `SD10` to earth. It is an error if the connection list is empty.

One of the limits of the current version of the simulator is that only the `Down` side of a circuit-breaker can be connected to a line. For instance, the following definition is forbidden, the penalty being very surprising results:

```
val L11 = line "L11" [(CB1,Up),(SD11,Up),
(SD10,Up)] 180.0 20.0 false;
```

Normal Configuration Once the devices and lines have been created, the normal network configuration must be initialised using the `set_normal_configuration` function which takes as parameters the list of devices and the list of lines in the network. For instance, for the rural network in Figure 1, we write:

```
set_normal_configuration
[CB1,CB2,CB3,CB4,CB5,CB6,CB7,SD10,SD11,SD12,
SD13,SD20,SD21,SD30,SD31,SD32,SD33,SD34,SD40,
SD41,SD42,SD43,SD44,SD50,SD51,SD52,SD53,SD60,
SD61,SD62,SD70,SD71,SD72]
[L10,L11,L12,L13,L14,L20,L21,L30,L31,L32,L33,
L34,L35,L40,L41,L42,L43,L50,L51,L52,L53,L60,
L61,L71,L70,L72];
```

It is an error if a device or line given as parameter has not been previously defined.

Faults At initialisation, all lines are assumed to be non-faulty. `set_faulty` takes as parameters a line and introduces a fault on this line. For instance :

```
set_faulty L12;
```

introduces a fault on `L12` and propagates the effect of this change onto the network (opens circuit-breakers feeding the fault and update powers entering the lines and devices affected).

Difficulty Level Recall there are two difficulty levels:

- level_1: no numerical power
- level_2 $(\beta, i_s, i_c, i_m, i_b)$: capacity constraints, and evaluation of plan cost using the simple cost model with parameters β, i_s, i_c, i_m, and i_b described above,
- level_3 (β, i_c, i_m, i_b): capacity constraints, and evaluation of plan cost using the sequential cost model with parameters β, i_c, i_m, and i_b described above.

The function `set_level` sets the difficulty level to the given one, for instance:

```
set_level (level_2 (3,1,5,2,3));
```

or again

```
set_level level_1;
```

Plan File

The format of the plan file is very simple. It just contains a plan described via the function `plan` which takes as parameters a list of (device, position) pairs and sets the devices to the given positions in the order specified. For instance:

```
plan [(SD11,Open),(CB1,Closed),
(SD12,Open),(SD53,Closed)];
```

It is an error if the devices have not been declared in the problem file. Naturally, at each step, the effects of the action executed are propagated through the network.

Simulation Report

Running the simulator produces a simulation report. This report indicates, at each step, which is the action performed, any circuit-breaker lost or recovered, any line lost or recovered, and, in difficulty level > 1, any change in the $pent$ power of a line or circuit-breaker. At the end of the simulation, the total cost of the plan is reported as well as its components, that is, in difficulty level 1, the number of lines non-supplied and the number of plan steps, in difficulty level 2, the number of plan steps, the number of critical lines not supplied, the breakdown costs, and the standard deviation of the margins of the circuit-breakers, and in level 3, the cumulative number of critical lines not supplied, the cumulative

breakdown costs, and the standard deviation of the margins of the circuit-breakers at the end of the plan.

Here is an example of report obtained:

```
------------------------------
network initialised
fault occurs on line L12
CB1, L10, L11, L12, L13, L14 are lost
------------------------------
step 1:
opening SD11
------------------------------
step 2:
closing CB1
CB1, L10, L11 are back
pent power change: CB1=25.0,
L10=5.0, L11=25.0
------------------------------
step 3:
opening SD12
------------------------------
step 4:
closing SD53
L13, L14 are back
pent power change: CB5=120.0, L13=20.0,
L14=40.0, L50=120.0, L51=110.0
------------------------------
plan valid
total cost: 1730.64527327
critical lines not supplied: 1
breakdown costs: 50.0
margin std: 13.9605859184
steps: 4
------------------------------
```

When a loop is created or, in difficulty level > 1, the capacities capacities of breakers or lines are exceeded, the simulator prints a message indicating this and aborts. The following is an example of a message obtained when the network specified in the problem file already violate the constraints:

```
capacity of CB1 exceeded
problem invalid -- aborting
```

And here is the message obtained in case a plan creates a loop:

```
the network has a loop
plan invalid -- aborting
```

Data

We provide problem examples for the rural network in Figure 1. These examples are partitioned into 3 directories (level1, level2, and level3) corresponding to the various difficulty levels.

For each of those, we provide a number of problems and plans. For instance, the `level2` directory contains:

- problem rural_network1 and plan rural_plan1
- problem rural_network2 and plans rural_plan2a, rural_plan2b, and rural_plan2c

The plan files contain the result of the simulation as a comment. Some of the plans are purposefully invalid.

Other Resources

The website: `http://users.rsise.anu.edu.au/~thiebaux/benchmarks/pds/` points to a number of additional useful resources and papers, such as the following.

Modelling: the website hosts the tool NET2PDDL which was used to produce the various PDDL formulations of the IPC-4 version of PSR. The formulations produced include one using derived predicates, another using conditional effects, and a pure STRIPS formulation. The following papers are useful to understand the problem and its modelling in PDDL: (Bertoli *et al.* 2002; Bonet & Thiébaux 2003; Hoffmann *et al.* 2006; Helmert 2006).

Data: the website contains some of the problem instances published in the literature, and RANDOMNET, the random generator used to produce the IPC-4 instances. Ideally, RANDOMNET would need to be extended to numerical powers for ICKEPS.

Credits

Our description of power distribution systems and of the supply restoration problem is based on work done in 1994-1996 in the framework of a contract between IRISA and the French electricity utility Electricité de France (EDF). We thank our collaborators at IRISA and EDF, in particular Marie-Odile Cordier, Isabelle Delouis-Jacob, Olivier Jehl and Jean-Paul Krivine. Many others have contributed to the development of PSR as a benchmark for planning, including, Blai Bonet, Piergiorgio Bertoli, Matt Gray, Charles Gretton, Jörg Hoffmann, Rune Jensen, and John Slaney.

References

Bertoli, P.; Cimatti, A.; Slaney, J.; and Thiébaux, S. 2002. Solving power supply restoration problems with planning via symbolic model-checking. In *Proc. 15th European Conference on Artificial Intelligence (ECAI-02)*, 576–580. Lyon (France): IOS Press.

Bonet, B., and Thiébaux, S. 2003. Gpt meets psr. In *13th International Conference on Automated Planning and Scheduling (ICAPS-03)*, 102–111. Trento, Italy: AAAI Press.

Helmert, M. 2006. New complexity results for classical planning benchmarks. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 52–61.

Hoffmann, J.; Eldekamp, S.; Thiébaux, S.; Englert, R.; Liporace, F.; and Trüg, S. 2006. Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4. *Journal of Artificial Intelligence Research* 26:453–541.

Thiébaux, S., and Cordier, M.-O. 2001. Supply restoration in power distribution systems — a benchmark for planning under uncertainty. In *Proc. 6th European Conference on Planning (ECP-01)*, 85–95.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of pddl axioms. *Artificial Intelligence* 168(1-2):36–69.