Power/Energy Track

Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads

Peter Palensky, Senior Member, IEEE, and Dietmar Dietrich, Senior Member IEEE

Contents

01 기존 전력망의 문제

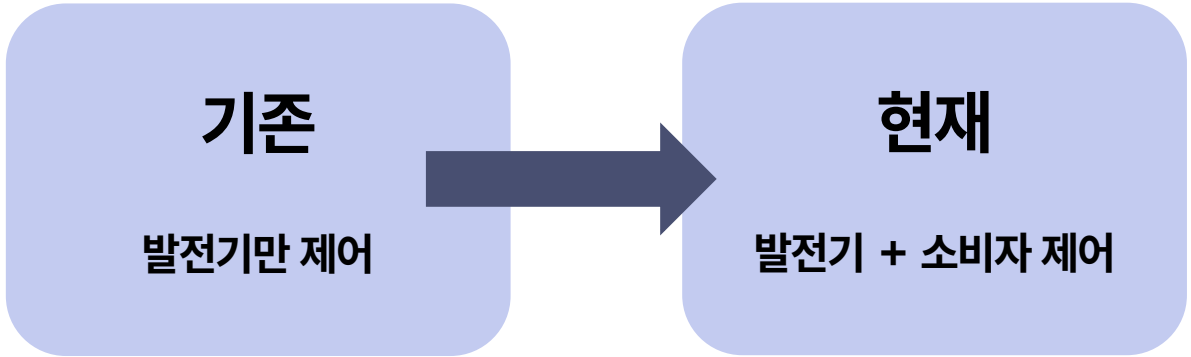
기존 전력망 : 발전량을 조절해서 수요를 맞추는 방식

출력 제어가 어려운 발전원 (태양광 발전, 풍력발전 , 전기차, ESS)의 증가

02 DSM (Demand Side Management)

기존 전력망 : 발전량을 조절해서 수요를 맞추는 방식

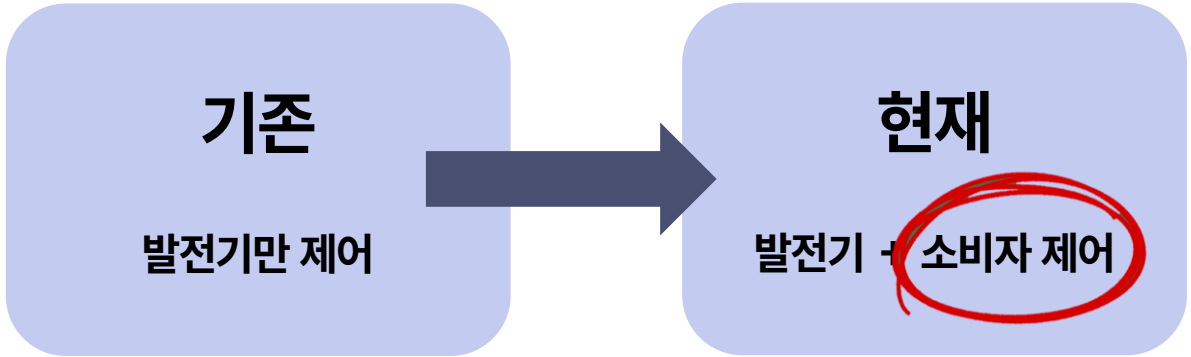
출력 제어가 어려운 발전원 (태양광 발전, 풍력발전 , 전기차, ESS)의 증가



02 DSM (Demand Side Management)

기존 전력망 : 발전량을 조절해서 수요를 맞추는 방식

출력 제어가 어려운 발전원 (태양광 발전, 풍력발전 , 전기차, ESS)의 증가



DSM (Demand Side Management)
= 수요측 관리, 전기를 사용하는 쪽의 전력 사용을 관리하는 기술

“ 전기를 사용하는 쪽에서 사용량이나 사용시간을 조절해 전력망을 효율적으로 운영하는 방법 ”

03 DSM - 4가지 분류

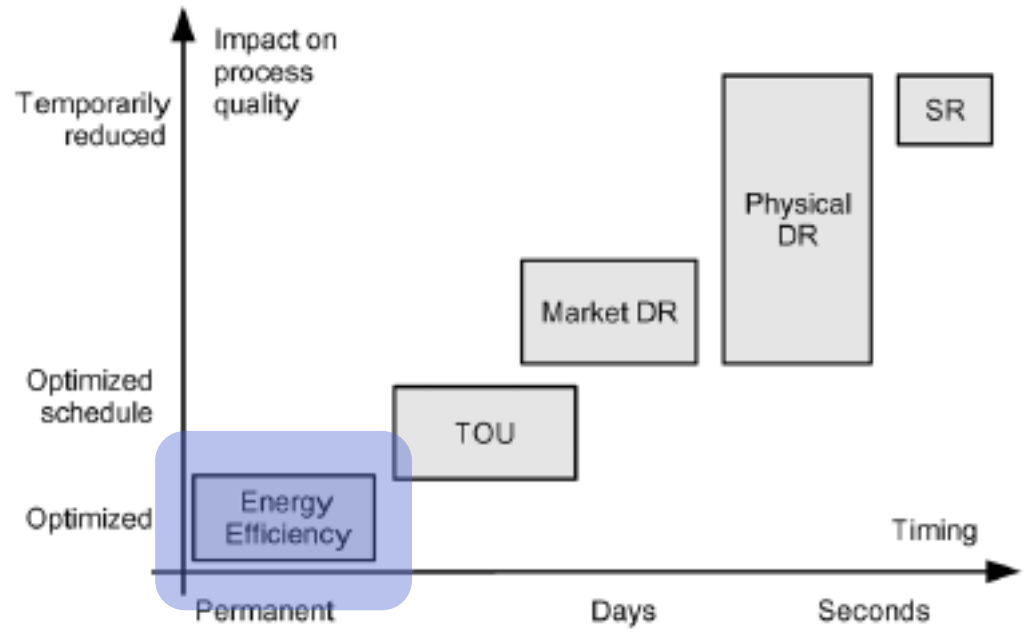


Fig. 1. Categories of DSM.

Energy Efficiency

: 전기를 영구적으로 적게 사용하는 것

- LED 조명 교체
- 고효율 모터 사용
- 건물 단열 개선
- 가장 느린 DSM

03 DSM - 4가지 분류

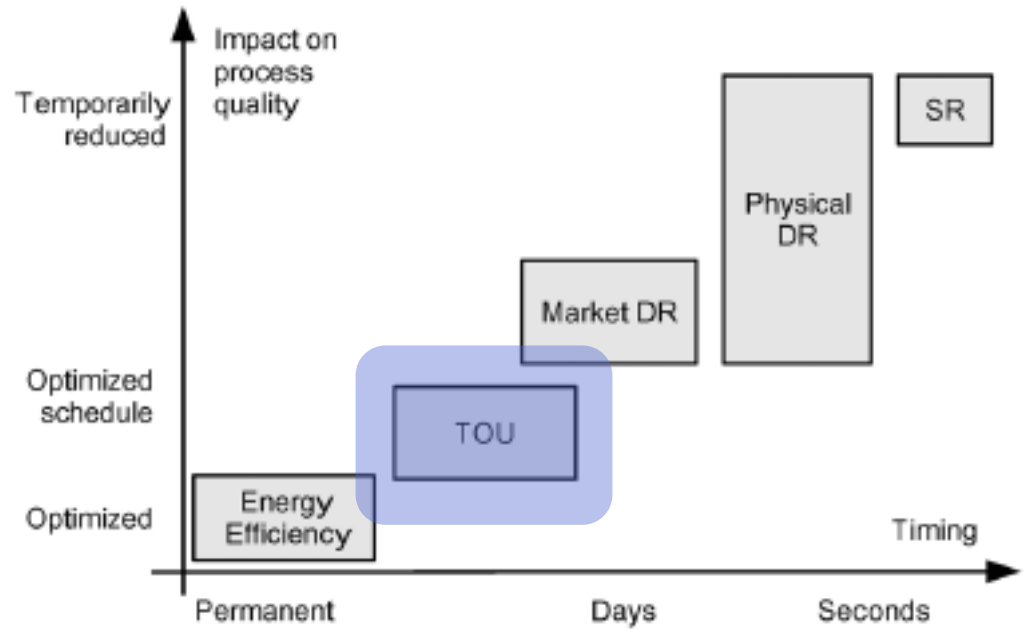


Fig. 1. Categories of DSM.

Time of Use

:전기를 덜 쓰는 것이 아니라 사용하는 시간을 옮기는 것.

- 세탁기를 오후 6이에 돌리지 않고 오전에 돌린다.
- 전기차 충전을 퇴근 직후가 아닌 새벽에 한다.

03 DSM - 4가지 분류

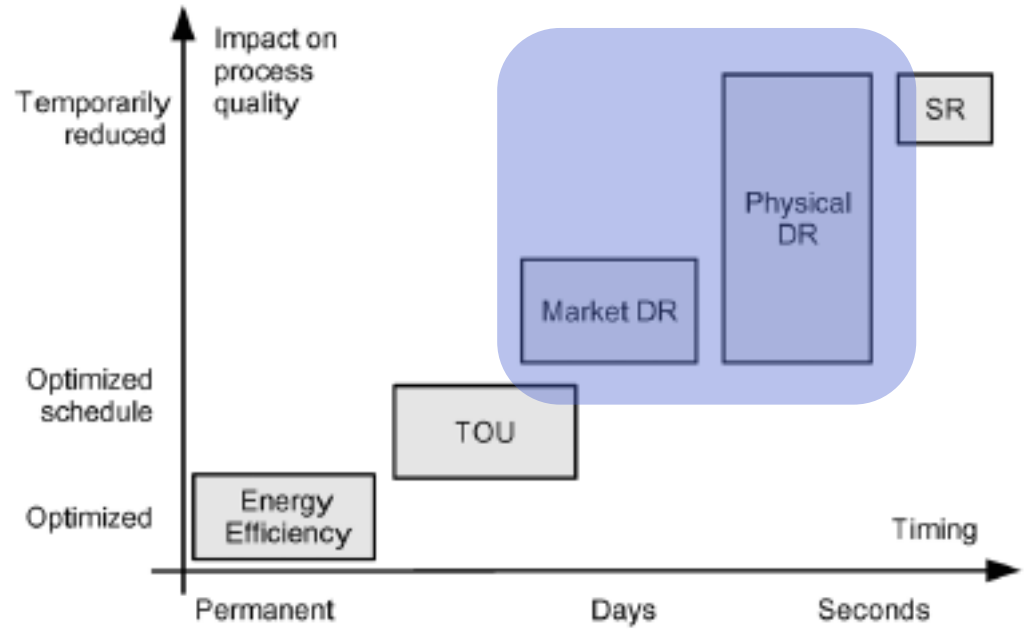


Fig. 1. Categories of DSM.

Demand Response : 수요 반응

전력회사 - "전기 소비 절감 요청"

공장 - 모터 일부 정지 , 냉동기 출력 감소

건물 – 에어컨 일부 OFF

03 DSM - 4가지 분류

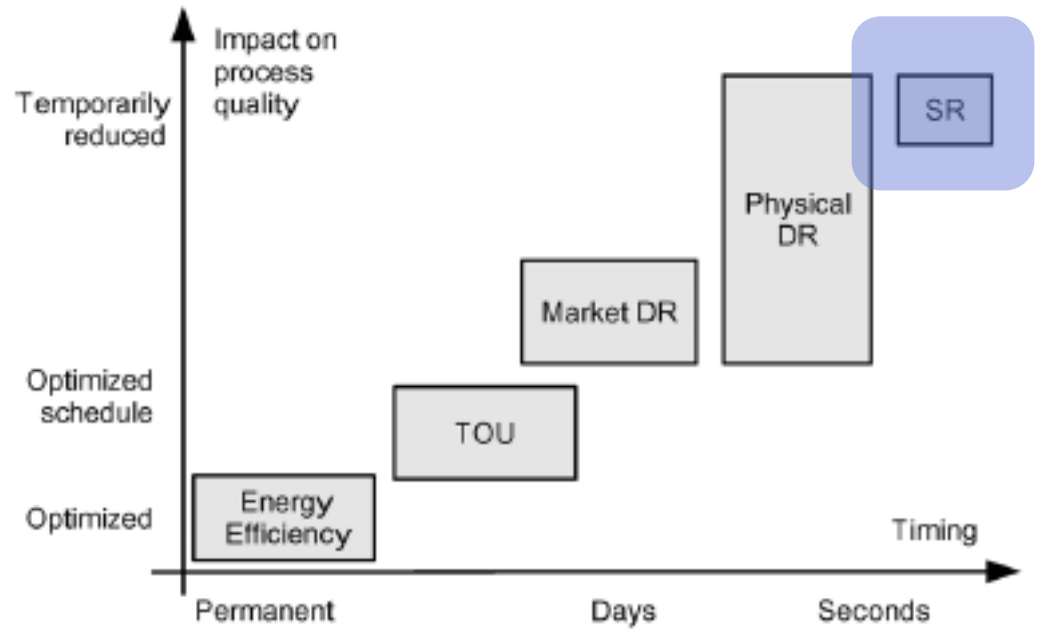


Fig. 1. Categories of DSM.

Spinning Response: 회전 예비력

주파수가 떨어지는 비상상황 발생

기존 발전기 측 관리 : 1차제어 -> 2차제어

DSM : 자동적으로 소비를 감소

03 추가설명

1

발전기에 작용하는 두 종류의 토크

- 터빈이 회전자를 돌리는 기계적 토크 Tm
- 발전기가 전력을 생산하면서 회전을 방해하는 전기적 토크 Te

2

정상상태 에서

$$T_m = T_e$$

3

$$P_e > P_m$$

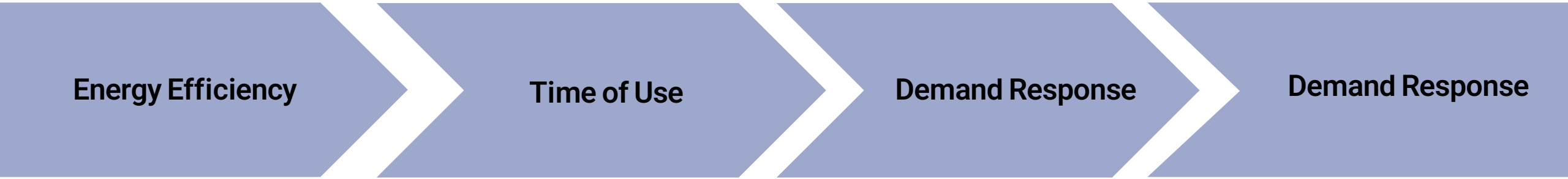
부하 증가 → 부족한 에너지를 채우기 위해 발전기 회전자에 저장되어 있던 운동에너지 사용 → 발전기 회전속도 감소

4

동기 발전기에서의 주파수와 회전속도 관계

$$n_s = \frac{120f}{P}$$

03 DSM - 4가지 분류



느린 수요 관리 → 빠른 수요 관리

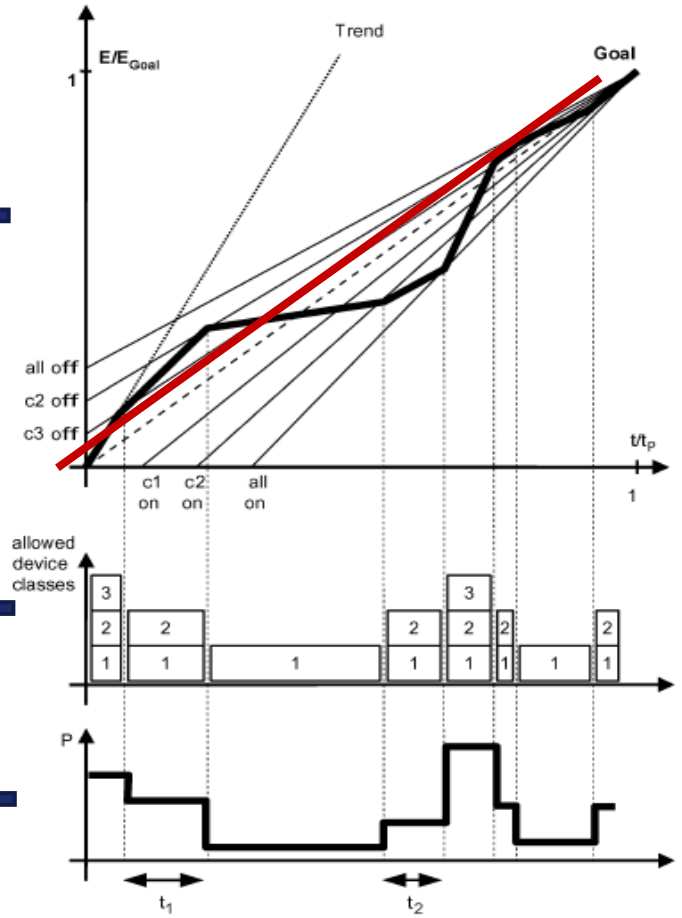
이 4가지의 DSM 방법은 각각 실행되는 것이 아닌,
EE를 기본으로 적용하고 TOU·DR·SR을 서로 다른 시간대의 제어 계층으로 결합하여 사용할 수 있다

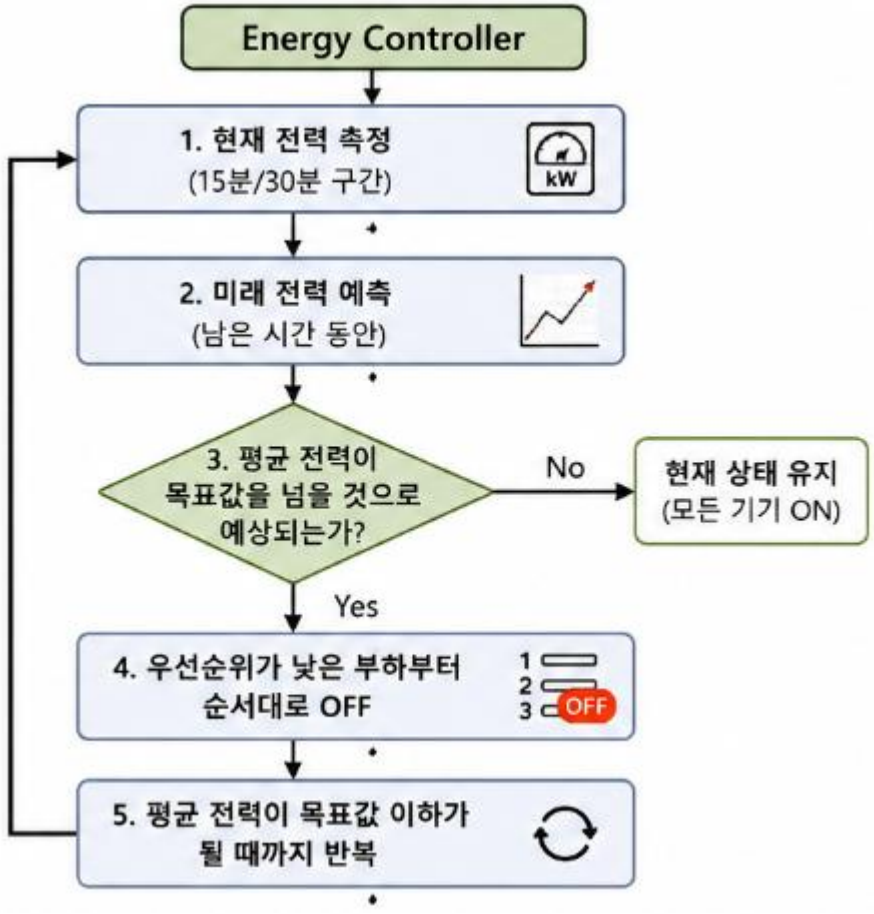
04 Energy Controller

누적 에너지 사용량

현재 작동하는 기기

현재 소비하는 전력

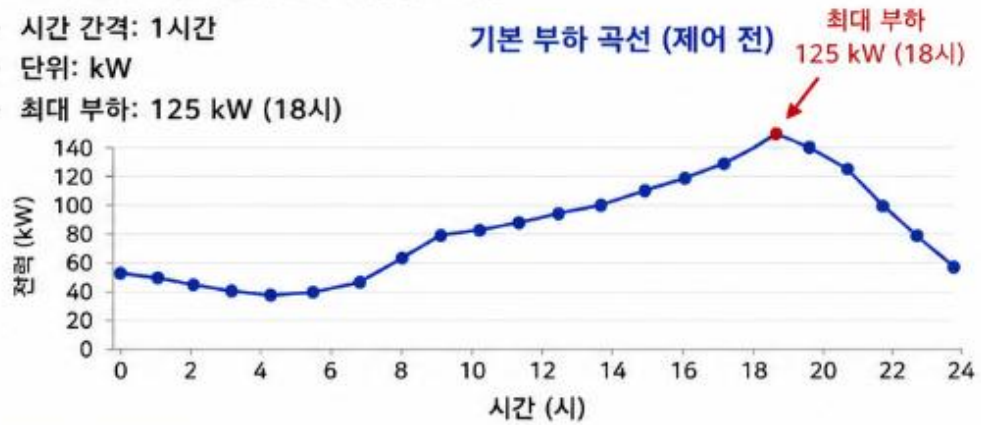




04 Energy Controller – 시뮬레이션 조건

1 기본 부하 데이터

- 24시간 건물의 시간별 전력 사용량을 가정
- 시간 간격: 1시간
- 단위: kW
- 최대 부하: 125 kW (18시)



2 목표 최대전력

목표 최대전력 (계약전력): **100 kW**

이 값을 초과하지 않도록 제어

3 제어 가능한 부하 목록

우선순위	부하 이름	소비전력 (kW)	설명
1	조명 일부	5	가장 먼저 차단
2	환기팬	8	두 번째 차단
3	전기차 충전	15	세 번째 차단
4	냉난방 일부	20	가장 나중에 차단

※ 숫자가 클수록 우선순위가 높아 먼저 차단됨

4 예측형 Energy Controller 알고리즘

예측 방식

현재 시점에서 앞으로 설정한 시간만큼 부하를 예측하여
예상 최대 부하가 목표 최대전력(100 kW)을 초과할 것으로
판단되면 미리 제어를 수행

부하 이동 규칙

- 차단된 부하는 최대 이동 가능 시간(설정값) 내에서 목표 최대전력을 초과하지 않는 가장 빠른 시간대로 이동
- 이동할 수 있는 시간이 없으면 해당 부하는 차단된 상태로 유지

5 기타 시뮬레이션 설정

예측 시간 범위 (forecast window)	: 2시간
최대 부하 이동 가능 시간 (max shift hour)	: 5시간
시간 간격	: 1시간
총 분석 시간	: 24시간 (0시 ~ 23시)

04 Energy Controller – 시뮬레이션 결과

Python 시뮬레이션 코드

```python
for hour in hours:
    cut_list = []

    # 앞으로 forecast_window 시간 동안의 최대 예상 부하
    future_end = min(hour + forecast_window + 1, 24)
    predicted_peak = max(controlled_load[hour:future_end])

    # 현재는 100kW 이하라도, 앞으로 2시간 안에 넘을 것 같으면 미리 제어
    if predicted_peak > P_limit:

        # 예측 구간 안에서 가장 부하가 큰 시간 찾기
        target_hour = hour + controlled_load[hour:future_end].index(predicted_peak)

        for device in controllable_loads:
            if controlled_load[target_hour] <= P_limit:
                break

            # target_hour에서 부하 차단
            controlled_load[target_hour] -= device["power"]
            cut_list.append(f"{device['name']}({target_hour}시)")

            # 차단된 부하를 뒤쪽의 여유 시간대로 이동
            moved = False
            for shift in range(1, max_shift_hour + 1):
                new_hour = target_hour + shift

                if new_hour >= 24:
                    break

                # 이동 후에도 목표 최대전력 이하이면 이동 허용
                if controlled_load[new_hour] + device["power"] <= P_limit:
                    controlled_load[new_hour] += device["power"]
                    shift_records.append({
                        "device": device["name"],
                        "power": device["power"],
                        "from": target_hour,
                        "to": new_hour
                    })
                    moved = True
                    break

            # 옮길 시간이 없으면 그냥 차단된 것으로 처리
            if not moved:
                shift_records.append({
                    "device": device["name"],
                    "power": device["power"],
                    "from": target_hour,
                    "to": None
                })
```

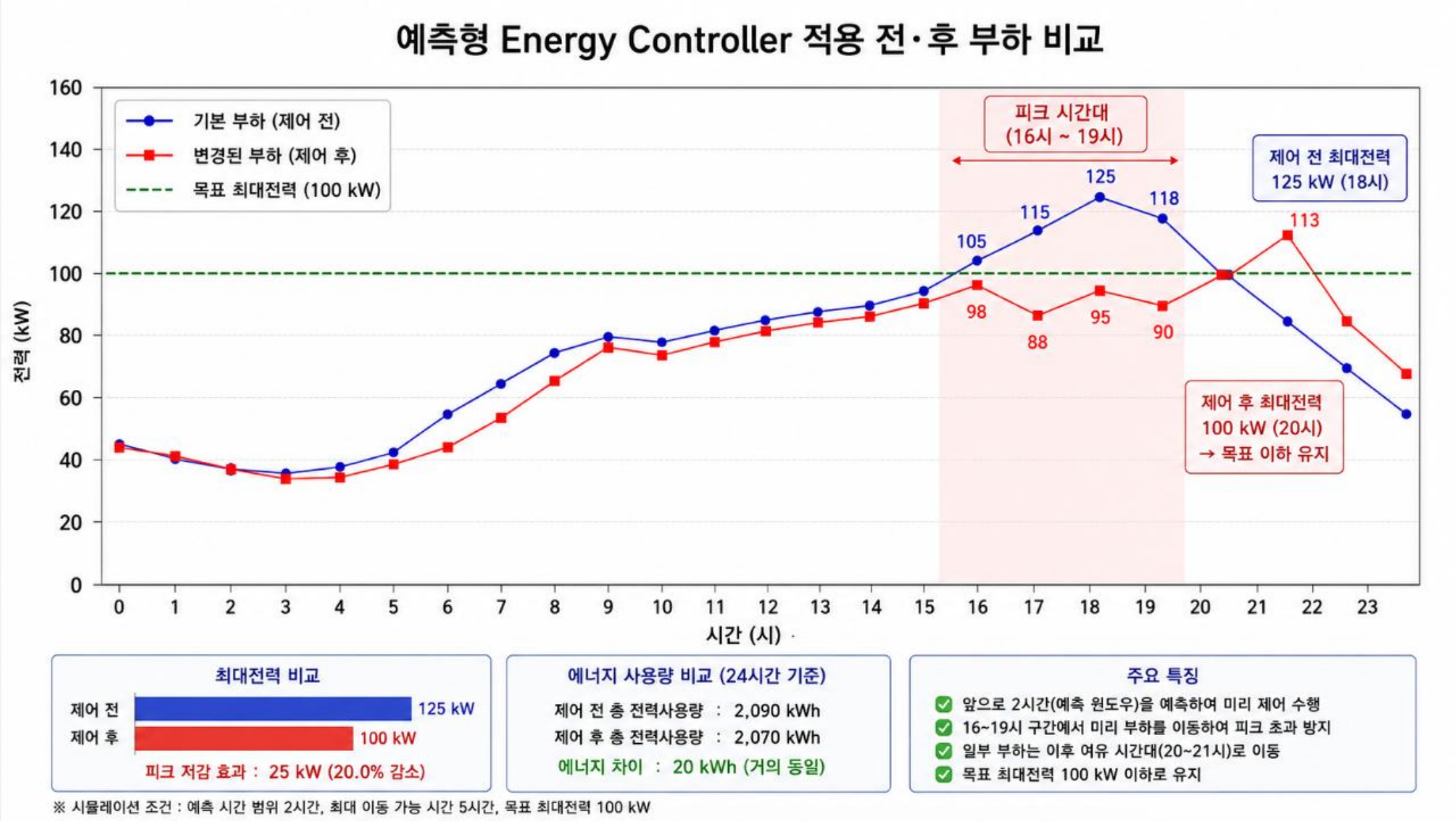
Python 시뮬레이션 결과

```
========== 예측형 Energy Controller 시뮬레이션 결과 ==========
목표 최대전력: 100 kW
예측 시간 범위: 앞으로 2시간
최대 부하 이동 가능 시간: 5시간
제어 전 최대전력: 125 kW
제어 후 최대전력: 100 kW
최대전력 감소량: 25 kW
최대전력 감소율: 20.0%
제어 전 총 전력사용량: 1812 kWh
제어 후 총 전력사용량: 1797 kWh
총 전력사용량 차이: 15 kWh

========== 예측 기반 제어 기록 ==========
14시 판단: 예측 피크 105 kW → 제어 실행: 조명 일부(16시)
15시 판단: 예측 피크 115 kW → 제어 실행: 조명 일부(17시), 환기팬(17시), 전기차 충전(17시)
16시 판단: 예측 피크 125 kW → 제어 실행: 조명 일부(18시), 환기팬(18시), 전기차 충전(18시)
17시 판단: 예측 피크 118 kW → 제어 실행: 조명 일부(19시), 환기팬(19시), 전기차 충전(19시)

========== 부하 이동 기록 ==========
조명 일부 5 kW: 16시 → 21시로 이동
조명 일부 5 kW: 17시 → 21시로 이동
환기팬 8 kW: 17시 → 22시로 이동
전기차 충전 15 kW: 17시 → 22시로 이동
조명 일부 5 kW: 18시 → 21시로 이동
환기팬 8 kW: 18시 → 23시로 이동
전기차 충전 15 kW: 18시 → 23시로 이동
조명 일부 5 kW: 19시 → 22시로 이동
환기팬 8 kW: 19시 → 23시로 이동
전기차 충전 15 kW: 19시에서 차단, 이동 가능한 시간 없음
```

04 Energy Controller – 시뮬레이션 결과



NOVA

Power/Energy Track

Thank you

숭실대학교 전기공학부 학술 소모임 NOVA
발표자 : 유나현