AI/ML

AI가 공장 일정을 짠다는 건 실제로 무슨 계산일까

방동걸 (주)웨이스 대표이사 2026년 8월 24일

"AI가 알아서 생산 일정을 짭니다." 자율형공장 이야기에서 가장 자주 나오는 문장이고, 가장 확인하기 어려운 문장이기도 합니다. 설비를 연결했다는 말은 대수를 세어 확인할 수 있고 불량을 잡았다는 말은 검사 결과로 확인할 수 있는데, 일정을 잘 짰다는 말은 무엇과 대조해야 할지가 애매합니다. 제조 AI 운영체제 VEXPLOR로 공장 일정을 짜는 웨이스가 실제로 무엇을 계산하는지 풀어 적었습니다.

그래서 그 안을 한번 뜯어 보겠습니다. 일정을 짜는 AI가 실제로 돌리는 계산은 무엇이고, 그 계산이 현장에서 어디서 깨지며, 깨지지 않게 하려고 무엇을 덧대야 하는지를 순서대로 적었습니다. 스케줄러를 검토하실 때 무엇을 물어봐야 하는지까지 마지막에 정리해 두었습니다.

01

목표가 하나면 계산이지만, 셋이면 선택입니다

먼저 짚어야 할 것이 있습니다. 생산 일정은 **정답이 하나 있는 문제가 아닙니다.**

공장이 일정에 걸고 있는 목표는 보통 최소 3가지입니다.

- **설비 가동률** — 기계를 놀리지 않는 것
- **납기 준수(리드타임)** — 주문을 약속한 날짜에 내보내는 것
- **에너지 비용** — 피크 시간대에 전력을 몰지 않는 것

그런데 이 셋은 서로를 밀어냅니다. 가동률을 끝까지 올리려면 같은 금형으로 찍을 수 있는 물량을 몰아서 돌리는 것이 유리한데, 그러면 뒤로 밀린 소량 주문의 납기가 깨집니다. 납기를 맞추려고 자주 갈아 끼우면 교체 시간만큼 가동률이 떨어집니다. 둘 다 잡으려고 야간에 라인을 더 돌리면 전력 단가가 올라갑니다.

그래서 **셋을 동시에 최고로 만드는 일정은 존재하지 않습니다.** 하나를 좋게 하면 다른 하나가 나빠지는 지점까지만 갈 수 있고, 거기서부터는 계산이 아니라 선택입니다. 이 상태를 다목적 최적화에서는 파레토 최적이라고 부릅니다. "더 이상 어느 하나도 남을 나쁘게 하지 않고는 좋아질 수 없는 상태"라는 뜻입니다.

이 이야기가 실무에서 왜 중요하냐면, **답을 하나만 내놓는 스케줄러는 이 선택을 이미 대신 해버린 것**이기 때문입니다. 어떤 가중치로 셋을 섞었는지 말해 주지 않으면, 그 일정이 왜 그렇게 나왔는지 아무도 설명할 수 없습니다.



가동률·납기·에너지가 같은 일정을 서로 다른 쪽으로 잡아당깁니다.

02

그래서 답을 하나가 아니라 여러 개 내놓습니다

여기서 유전 알고리즘 계열을 사용합니다. 대표적인 NSGA-II 같은 다목적 최적화 알고리즘은 일정 후보를 한 무더기 만들어 놓고, 서로 비교해 가며 세대를 반복합니다.

이 방식은 **답을 하나로 줄이지 않습니다.** 결과물은 "최적 일정 1개"가 아니라 서로 우열을 가릴 수 없는 후보 집합입니다.

- **후보 A** — 가동률이 높지만 소량 주문 3건이 납기를 넘깁니다
- **후보 B** — 납기를 전부 지키지만 금형 교체가 늘어 가동률이 내려갑니다
- **후보 C** — 둘의 중간이지만 야간 전력 구간에 걸칩니다

세 후보 중 무엇을 고를지는 그날 공장의 사정입니다. 이번 달 납기 사고가 한 번 났으면 B고, 전력 계약 상한이 걸려 있으면 C입니다. **알고리즘은 선택지를 만들고, 선택은 사람이나 상위 규칙이 합니다.** 스케줄러를 도입할 때 "왜 이 일정이냐"에 답할 수 있는지가 여기서 갈립니다.

03

수학적으로 완벽한 일정이 현장에서는 실행이 안 됩니다

여기까지는 교과서대로입니다. 실제로 만들어 보면 여기서부터 막힙니다.

최적화가 내놓은 일정 중 상당수가 **물리적으로 실행 불가능**합니다. 숫자로는 가장 좋은데, 현장에 가져가면 그런 순서로 돌릴 수가 없습니다.

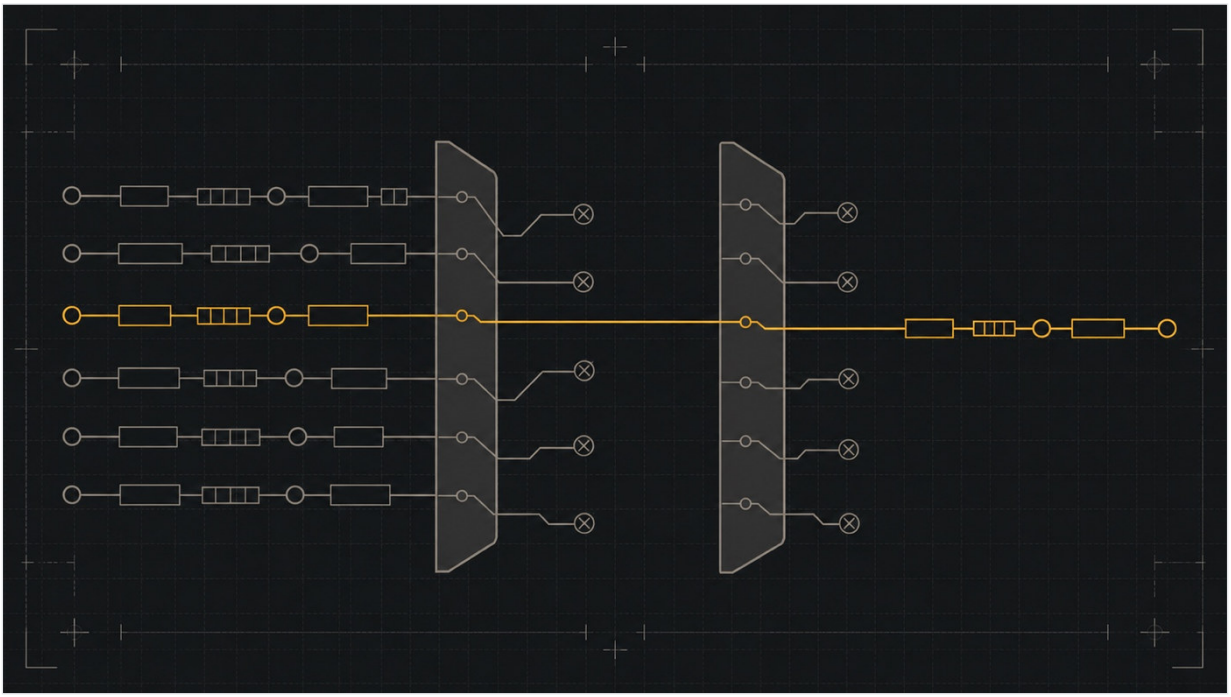
- **자재가 그 시각에 없습니다.** BOM을 따라가면 하위 부품이 도착하는 시각이 정해져 있는데, 상위 공정을 그보다 앞에 배치해 버립니다.
- **같은 설비를 두 작업이 동시에 잡습니다.** 후보를 무작위로 조합하다 보면 자원 충돌이 그대로 남습니다.
- **금형·치공구가 하나뿐입니다.** 같은 금형을 쓰는 두 작업을 겹쳐 놓으면 계산상으로는 리드타임이 줄지만 현장에서는 성립하지 않습니다.

이걸 처리하는 방법은 크게 2가지입니다.

하나는 패널티 함수입니다. 제약을 어긴 후보에 큰 벌점을 매겨서 세대를 반복하는 동안 자연스럽게 도태되게 만듭니다. 구현이 간단하고 제약 종류가 늘어도 붙이기 쉽습니다. 대신 벌점 크기를 잘못 잡으면 "조금 어긴 좋은 답"이 살아남아 결국 실행 불가능한 일정이 나옵니다.

다른 하나는 제약 만족 문제(CSP)로 푸는 방법입니다. 각 작업에 시작 가능한 시간 창을 먼저 계산해 놓고, 그 창 안에서만 배치하다가 막히면 되돌아가서 다시 시도합니다. 백트래킹이라고 부르는 방식입니다. 실행 불가능한 답이 애초에 만들어지지 않습니다. 대신 제약이 촘촘해질수록 계산이 급격히 무거워집니다.

실제 설계에서는 둘을 섞습니다. **깨지면 안 되는 제약은 시간 창으로 먼저 잘라내고, 지키면 좋은 조건은 벌점으로 넣습니다.** 자재 도착 시각이나 설비 중복은 전자고, 금형 교체 횟수를 줄이는 것은 후자입니다. 이 구분을 처음에 제대로 해 두지 않으면 "왜 실행 못 하는 일정이 나오느냐"는 질문을 끝까지 받습니다.



후보 일정이 검증 2단을 지나며 걸러지고, 실행 가능한 하나만 남습니다.

04

학습한 모델이 처음 보는 상황을 만나면 어떻게 될까요

일정 전체를 한 번에 짜는 일과, 지금 이 설비가 다음에 무엇을 집을지 정하는 일은 다릅니다. 후자를 디스패칭이라고 부릅니다. 여기에는 강화학습(DQN·PPO 계열)을 적용합니다. 현재 대기열 상태를 보고 다음 작업을 고르는 정책을 시뮬레이션 안에서 반복 학습시키는 방식입니다.

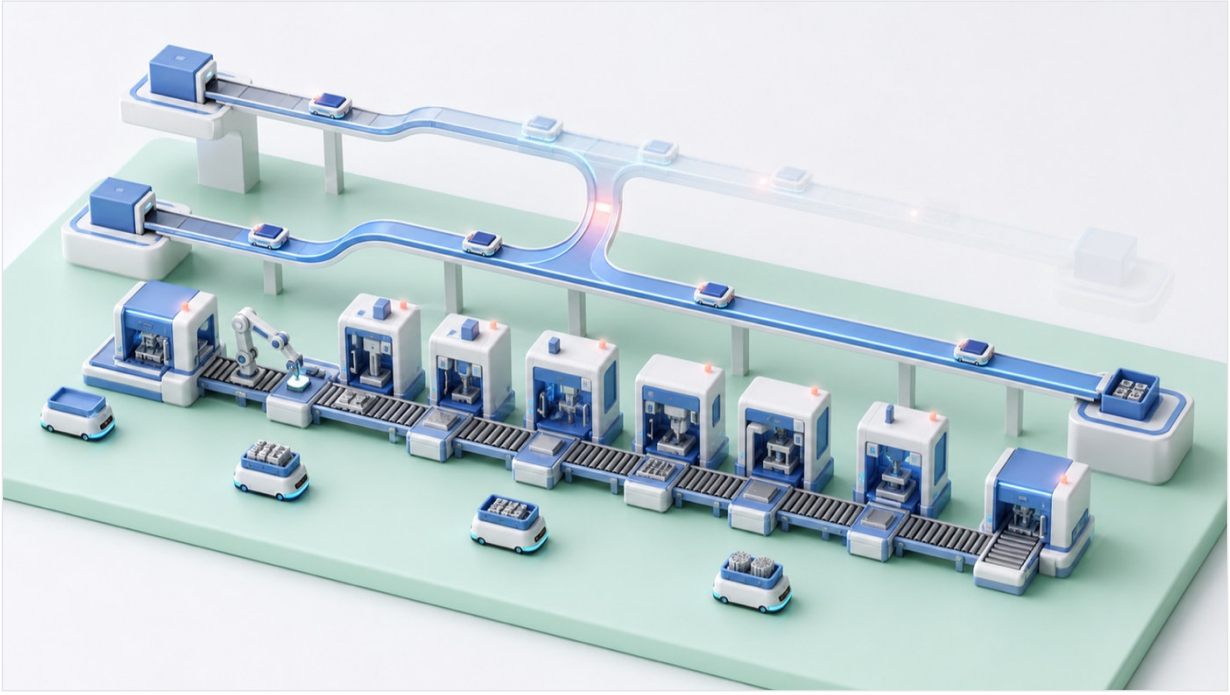
잘 학습되면 사람이 만든 규칙보다 낫습니다. 그런데 **학습한 적 없는 상태가 들어오면 달라집니다.** 갑작스러운 대량 긴급 주문, 설비 2대 동시 고장, 평소 안 쓰던 자재 조합처럼 학습 분포 밖의 상황에서 강화학습 정책은 조용히 이상한 선택을 합니다. 오류를 내지 않고 그냥 나쁜 답을 자신 있게 내놓기 때문에 더 위험합니다.

그래서 학습 정책만 두지 않고 **고전 규칙을 함께 둡니다.**

규칙	기준	강한 상황
SPT	짧은 작업 먼저	대기열 밀림
EDD	납기 빠른 것 먼저	납기 사고
CR	여유 시간 비율	혼재 주문
ATCS	납기와 교체 절충	금형 교체 많음

이 규칙들은 수십 년 됐고 최적도 아니지만, **상황을 가리지 않고 설명 가능한 답을 냅니다.** 학습 정책의 확신도가 낮거나 입력이 학습 범위를 벗어났다고 판단되면 규칙 쪽으로 내려오게 설계합니다. 성능의 최고점은 낮아지지만 최저점이 올라갑니다. 공장에서는 후자가 더 중요합니다.

우리도 자율형공장 과제에서 스케줄링을 설계할 때 이 구조를 먼저 잡았습니다. **잘 될 때 얼마나 잘 되는가보다, 안 될 때 얼마나 안전하게 실패하는가를 기준으로 정했습니다.** 안 될 때의 바닥이 정해져 있지 않으면 현장에서 그 기능을 아무도 켜 두지 않기 때문입니다.



학습 정책이 확신을 잃으면 위 경로가 끊기고 아래의 단순한 규칙 경로가 이어받습니다.

05

내보내기 전에 가상에서 한 번 돌려 봅니다

일정이 나왔다고 바로 작업지시로 내보내지 않습니다. 그 사이에 검증 단계를 하나 둡니다.

만들어진 일정을 시뮬레이션에 넣고 공정 전체를 다시 계산합니다. 여기에 두 방식을 함께 넣는데, 성격이 다릅니다.

- **이산사건 시뮬레이션(DES)** — 공정을 사건의 연속으로 봅니다. 언제 시작하고 언제 끝나고 언제 대기가 생기는지를 계산합니다. 반복 공정의 시간·처리량 예측에 강합니다.
- **에이전트 기반 시뮬레이션(ABS)** — 물류 로봇이나 작업자처럼 각자 판단하는 대상을 개별 객체로 놓고 상호작용을 계산합니다. 경로 충돌이나 병목처럼 개별 행동이 모여 생기는 현상을 봅니다.

둘을 같이 쓰는 이유는 공장에 두 종류의 문제가 동시에 있기 때문입니다. 프레스 라인의 시간 계산은 DES가 맞고, 그 사이를 오가는 로봇의 경로 간섭은 ABS가 맞습니다. 하나만 쓰면 다른 쪽 문제가 시뮬레이션에서 보이지 않습니다.

여기서 반드시 확인해야 할 것이 하나 있습니다. **그 시뮬레이션이 실제와 얼마나 맞는지를 숫자로 재 두었느냐**입니다. 가상에서 나온 결과가 현장에서도 나오는지를 실제 생산 값과 대조해 보지 않았다면, 그 검증 단계는 통과 도장에 지나지 않습니다. 우리는 자율형공장 1호 현장에서 시간·자원·생산량 3가지를 종합한 산식으로 정확도 85%를 확인했습니다(스피폭스 실증 기준). 공인 인증은 2027년 예정이라 아직 받지 않았습니다.

06

스케줄러를 검토하실 때 물어보실 것

지금까지 적은 내용을 확인 질문으로 바꾸면 이렇습니다. 도입을 검토하는 자리에서 그대로 쓰실 수 있게 정리했습니다.

- **목표를 몇 개 놓고 계산합니까.** 하나면 나머지는 어디로 갔는지, 여럿이면 가중치를 누가 정하는지 물어보십시오.
- **답을 하나만 줍니까, 여러 개 줍니까.** 하나만 준다면 선택을 이미 대신 한 것입니다.
- **실행 불가능한 일정을 어떻게 걸러냅니까.** 벌점인지 시간 창인지, 자재 도착과 설비 중복은 어느 쪽으로 처리하는지 확인하십시오.
- **학습 범위 밖 상황에서는 무엇으로 넘어갑니까.** 규칙 풀백이 없으면 그 기능은 결국 꺼진 채로 남습니다.
- **일정을 내보내기 전에 무엇으로 검증합니까.** 시뮬레이션을 쓴다면 그 시뮬레이션의 정확도를 무엇과 대조해 쟀는지까지 물어보십시오.

마지막 질문이 특히 중요합니다. 앞의 4가지는 설계 문서를 보면 확인할 수 있지만, 검증 정확도는 실제 라인 데이터와 맞춰 본 회사만 대답할 수 있습니다.

07

정리하면

공장 일정을 AI가 짠다는 말은 기술 하나를 가리키지 않습니다. **여러 계산을 순서대로 이어 놓은 구조**를 말합니다. 목표가 여럿이라 후보를 여러 개 만들고, 그중 실행 불가능한 것을 제약으로 걸러내고, 순간 판단은 학습 정책이 하되 범위를 벗어나면 규칙으로 내려오고, 최종 일정은 가상에서 한 번 돌려 본 뒤에 내보냅니다.

이 순서 중 어느 하나가 비어 있으면 나머지가 아무리 정교해도 현장에서는 쓰이지 않습니다. 계산이 틀려서가 아니라, **틀렸을 때 어떻게 되는지가 정해져 있지 않아서** 아무도 켜 두지 않기 때문입니다.

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.

원문 blog.vexplor.com/post/factory-scheduling-what-ai-computes
발행 2026년 8월 24일 · 방동걸 (주)웨이스 대표이사

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
연락처 전화 070-4730-0629 · contact@wace.me
홈페이지 wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인