배터리 잔량에 따른
효율적 스케줄링 시스템

- 알고리즘 소개 -

Team Eco_BARAM

임베디드 SW 경진대회
스마트팩토리 부문

배터리 잔량에 따른 업무부여 알고리즘

들어가기 전에, 본 장에서는 많은 가정을 기반으로 진행된 부분들이 존재한다. 이 가정의 대부분은 실제와 유사한 상황에서 사용되었거나 예시를 들기 위한 목적으로 사용되었다. 가정의 대부분을 제거하여도 본 알고리즘의 작동에 문제가 없음을 알린다.

알고리즘 전략의 배경

알고리즘을 적용하고자 하는 공장마다 요구하는 목표가 다를 것이다. 어떤 공장은 하루 동안의 업무 처리량만이 중요할 것이다. 반면 어떤 공장은 커다란 생산 라인의 일부라서 일정량 이상의 업무 처리량을 꾸준히 내지 못하면 라인 전체에 영향을 끼치게 된다. 즉 매 순간 처리되어야 하는 최소 업무 처리량이 존재한다.

매 순간 구동되었으면 하는 로봇의 최소 수가 존재하고, 이는 적용하고자 하는 **공장마다 다르다**는 것이다. 그렇기에 우리는 업무 처리중인 로봇의 최소 수를 정한 상태에서 업무 처리량을 최대화하고자 한다. (단 이 로봇의 최소 수에 어떤 값이 오더라도 적용 가능한 알고리즘을 찾을 것이다.)

업무 처리량을 증가시키기 위해서 할 수 있는 조치가 세 가지 있다.

①일을 부여하기에 가장 적절한 로봇을 찾는다.

②로봇의 동선을 최적화한다.

③로봇이 충전소를 자주 드나들지 않는다.

이 중 ②번은 앞서 소개한 BFS algorithm과 making route algorithm을 통해 해결되었다. 또한 이번 항목에서 ①번 조치를 시행할 예정이다.(뒷부분에서 설명) 그렇다면 공장이 하루동안 처리하는 일의 양을 늘리기 위해서는 로봇이 충전소를 자주 드나들지 않는 방법밖에 남지 않았다.

즉 본 알고리즘은 일을 처리하고 있는 로봇의 최소 수를 고정해두고 각 로봇이 충전소에 가장 오래 머물 수 있도록 하는 방법을 제시한다. (본 알고리즘은 성능이 좋은 해를 제시하지만 최적해를 제시하지는 않음을 밝힌다.)

- 로봇과 충전기의 수 설정

우선 로봇의 수와 충전기의 수를 어떻게 설정해야 하는지를 알아본다. 충전기의 수가 부족하거나 로봇의 수가 너무 많다면 전체 배터리의 총량은 점차 감소할 것이고, 충전하지도 일하지도 않는 로봇이 필연적으로 발생하여 비효율적이게 된다. 즉 모든 충전기를 로봇들이 사용하고 있을 때 충전중인 로봇들의 충전량의 합은 일하는 로봇들이 소모하는 소모량의 합보다 크거나 같아야 한다. 이를 식으로 나타내면 아래와 같다.

※ (N-N_{chrg})C_{max} <= N_{chrg}*Chrg

N : 로봇의 수 / N_{chrg} : 충전기 수 / C_{max} : 시간당 배터리 소모량 / Chrg : 시간당 배터리 충전량

이 조건은 전체 배터리의 총량이 점점 감소하지 않도록 한다.

이 조건을 만족하지 않을 경우 모든 충전기를 사용하더라도 전체 배터리의 총량은 감소한다.

즉 충전과 작업 중 어느 하나도 하지 않게 되는 로봇이 존재하게 된다.

위의 식을 만족하지 않는 예시)

N=4, N_{chrg}=1, C_{max}=12, Chrg = 30

로봇 4대의 초기 배터리량이 100, 100, 100, 100이었다고 생각해보자. (배터리 총합=400)

1대를 충전하고 3대를 작동시키면 130, 88, 88, 88이 된다.

다음 로봇을 충전시키면 118, 118, 76, 76이 된다.

다음 로봇을 충전시키면 106, 106, 106, 64가 된다.

다음 로봇을 충전시키면 94, 94, 94, 94가 된다. (배터리 총합=376)

모든 시간에 모든 충전기를 가동하였음에도 불구하고 배터리 총합이 감소했다.

- '일을 처리하고 있는 로봇의 최소 수'와 충전기 수의 상관관계

일을 처리하고 있는 로봇의 수가 최소라는 것은 충전기를 사용하고 있거나 사용하려는 로봇의 수가 최대라는 것이다. 이는 모든 충전기를 사용하고 있거나 사용하러 가고 있음을 의미한다.

일을 처리하고 있는 로봇의 최소 수 <= 전체 로봇의 수 N – 충전기 수

즉 일을 처리하고 있는 로봇의 최소 수를 결정하기 위해서는 충전기 수를 결정하면 된다. 직전의 절에서 전체 로봇 수와 충전기 수를 결정하였으니 일을 처리하고 있는 로봇의 최소 수의 범위도 결정이 되었다.

- 배터리 <-> 이동 거리 환산

로봇의 배터리 잔량은 이동 가능한 거리로 환산이 가능하다. 로봇이 움직일 때 초당 소모하는 배터리 퍼센티지를 C[%/s]로, 로봇의 이동 속도를 V[m/s]로, 이동하는 거리를 Dist[m]로, 로봇의 배터리를 B[%]로 두면 배터리와 거리의 관계는 다음과 같다.

$$\mathrm{B} = \mathrm{Dist} \times \frac{C}{V}$$

이 때 로봇의 이동 속도의 목표치는 항상 같다. 가속과 감속 시에는 속도가 변하지만, 전체 이동하는 시간이 가감속 시간보다 많이 기므로 V는 항상 일정하다고 본다. 또한 움직일 때 초당 소모하는 배터리도 모터 온도, 차체가 든 짐의 무게 등에 따라 변할 수 있지만 이 값 또한 일정하다고 본다.

즉 배터리와 이동 가능한 거리는 상수를 곱하여 환산 가능한 관계이다.

- "여분의 이동 가능 거리값" 설정

"여분의 이동 가능 거리값"이라 불리는 값을 중심으로 알고리즘을 만들었다. 이 절에서는 해당 값에 대해 알아본다.

한 로봇의 배터리 잔여량을 $\mathrm{B_{total}}$이라 둔다. 또한 하던 일을 마무리한 후 충전소로 복귀하기까지 필요한 배터리의 양을 $\mathrm{B_{min}}$이라 둔다. $\mathrm{B_{total}}$에서 $\mathrm{B_{min}}$만큼의 배터리를 남긴 나머지 배터리의 양을 B_{extra}라 둔다. 이를 식으로 나타내면 다음과 같다.

$$B_{total} = B_{extra} + \mathrm{B_{min}}$$

이제 한 로봇의 배터리는 일을 마치고 충전소로 복귀하기 위한 배터리 값과 여분의 배터리 값으로 나뉘게 되었다.

직전의 절에서 배터리와 거리는 환산 가능함을 보였다. 위 식을 거리로 표현하면 다음과 같다.

$$Dist_{total} \cdot \frac{C}{V} = Dist_{extra} \cdot \frac{C}{V} + \mathrm{Dist_{min}} \cdot \frac{C}{V}$$

양 변을 $\frac{C}{V}$로 나눈다.

$$Dist_{total} = Dist_{extra} + \mathrm{Dist_{min}}$$

앞으로 $Dist_{total}$변수는 거의 등장하지 않을 것이고, $Dist_{extra}$ 변수와 $Dist_{min}$변수를 중심으로 알고리즘을 서술할 것이다.

- 시스템의 전제

모든 로봇은 시간당 배터리 충전량이 같다고 가정한다.

모든 로봇은 시간당 배터리 소모량이 같다고 가정한다. (단, 같지 않아도 시스템이 유지되도록 설계되었다. 다만 이에 따른 업무 처리량의 감소가 얼마나 될 지는 알 수 없다.)

시간당 배터리 충전량이 시간당 배터리 소모량보다 크다고 가정한다.

알고리즘 전략의 수립

- 로봇의 상태

모든 로봇은 충전 중이거나, 충전하러 가고 있거나, 일을 하는 중이거나, 일을 끝낸 후 다음 동작에 대한 명령을 기다리는 4가지 상태 중 하나를 가진다.

- 상태 변화는 최소화

로봇이 충전소를 자주 드나들면 충전소를 오가는 배터리와 시간이 낭비된다. 그러므로 일하는 중인 로봇은 가능한 한 충전하지 않도록 하고, 충전중인 로봇은 업무 효율성을 해치지 않는 한 일하지 않도록 한다.

- 시스템의 정의

로봇이 충전중인 경우 $\mathrm{Dist}_{\min}$는 0이고 $Dist_{extra}$는 시간당 배터리 충전량에 비례하여 증가한다. 로봇이 충전소로 이동중인 경우 $\mathrm{Dist}_{\min}$는 시간당 배터리 소모량에 비례하여 감소한다. $Dist_{extra}$는 유지된다.

로봇이 일을 하는 중인 경우 $\mathrm{Dist}_{\min}$는 시간당 배터리 소모량에 비례하여 감소한다. $Dist_{extra}$는 유지된다.

- 최소 일꾼 수 유지

일을 처리하고 있는 로봇의 최소 수를 설정한 후 업무 처리량을 극대화하는 것이 우리의 목적임

을 알고리즘 전략의 배경에서 밝혔다. 이 절에서는 일을 하는 중인 로봇의 최소 수를 유지하는 방법에 대해 제시한다.

일을 하는 중인 로봇의 최소 수를 Nmin이라 하겠다.

Nmin보다 큰 수의 로봇이 일을 하고 있는 경우 : 일을 끝낸 후 다음 동작에 대한 명령을 기다리는 로봇에게 계속 일을 준다.

Nmin만큼의 로봇이 일을 하고 있는 경우 : 일을 끝낸 후 다음 동작에 대한 명령을 기다리는 로봇에게 해당 로봇의 배터리 양에 따라 충전을 시킬지, 일을 줄지 결정한다. 만약 충전을 시킨다면 충전중인 로봇 중 가장 배터리가 많은 로봇에게 일을 부여한다. (배터리 양과 충전 여부의 관계는 다음 절에 설명)

- 일하는 중인 로봇의 수가 Nmin일 때 일을 끝낸 로봇의 충전여부 결정

일을 하는 로봇은 계속 일을 하도록 해야 업무 처리량을 늘릴 수 있음을 '상태 변화는 최소화' 절에서 말했다. 즉 해당 로봇의 배터리가 새로운 일을 받아도 될 만큼 넉넉하다면 해당 로봇에게 새로운 일을 부여해도 될 것처럼 생각된다. 이를 수식으로 나타내면 아래와 같다.

$$\text{if}(Dist_{extra} - Dist_{NewWork} \geq 0)\text{이면 새로운 일 부여}$$

※ 문제점 : 다음의 경우에 문제가 생길 수 있다.

모든 로봇이 비슷한 수준의 배터리를 가진 상태에서 일을 하고 있고, 비슷한 시점에 현재의 일이 끝나는 상황을 생각해보자. 새로운 일에 소요되는 이동거리 $Dist_{NewWork}$는 로봇마다, 일마다 다르지만 어느 정도 범위 내에 있다. 지금은 $Dist_{NewWork}$가 일정하다고 가정하자. 또한 로봇의 수는 충전기 수의 2배 정도라고 가정한다.

$Dist_{extra}$가 $Dist_{NewWork}$보다 클 때는 문제가 발생하지 않는다. 하지만 일을 한번 부여받을 때마다 로봇의 $Dist_{extra}$는 $Dist_{NewWork}$만큼 줄어든다. $Dist_{extra}$가 $Dist_{NewWork}$보다 작은 상태가 되면 로봇의 충전 여부를 결정해야 하는 상황이 온다. 각 로봇의 초기 배터리와 $Dist_{NewWork}$가 같다고 가정한 상황이므로 충전 여부를 결정해야 하는 상황은 거의 비슷한 시점에 일어날 것이다. 이렇게 되면 이 로봇들은 한꺼번에 충전기로 몰려가려 한다.

조금이라도 빨리 일을 끝낸 로봇은 충전기로 향하여 모든 충전기를 먼저 채운다. 하지만 굉장히 짧은 시간이 지나면 다른 로봇들이 일을 끝내고 충전기 자리를 내놓으라고 요청하게 된다. 충전기에 먼저 도착한 로봇들은 찰나의 충전을 마치고 충전기에서 나온다. 하지만 새로운 일을 부여받을 수는 없다. $Dist_{extra}$가 $Dist_{NewWork}$보다 여전히 작기 때문이다. 충전기에서 나오자마자 충

전 대기 상태에 돌입하고, 충전 대기 상태인 로봇들보다 충전기를 점유한 로봇들의 배터리가 조금이라도 높아지면 충전기에 있는 로봇들과 충전 대기중인 로봇은 또다시 교체하게 된다.

굉장히 오랜 시간동안 두 무리의 로봇들이 서로 바톤 터치를 하고 있는 상황이 되어버린 것이다. 이러한 루프를 탈출하기 위해서는 위의 수식을 바꾸어야 한다. 기존 수식은 아래와 같았다.

$$\text{if}(Dist_{extra} \geq Dist_{NewWork})\text{이면 새로운 일 부여}$$

※ 해결책 :

모든 로봇들이 새로운 일 한번에 대한 소요 거리에 해당하는 배터리를 비축하고 있어서 위와 같은 상황이 생겼다. 로봇들을 $Dist_{extra}$에 대한 오름차순으로 정렬한 후 배터리를 상대적으로 많이 가진 로봇들은 가져야 하는 최소 배터리량을 $Dist_{NewWork}$의 정수배를 적절히 더해준다. 새로운 수식은 아래와 같다.

$$\text{if}(Dist_{extra} >= \left\lfloor \frac{i-1}{\text{충전기 수}} \right\rfloor Dist_{NewWork})\text{이면 새로운 일 부여}$$

로봇들을 배터리가 작은 순으로 정렬했을 때 앞에서부터 i번째 로봇에 해당하는 수식이다. 꺾인 괄호는 버림 표시이다. 위의 계산을 다시 한번 거쳐보면 무한루프가 생기지 않음을 확인할 수 있다. 계산은 복잡하여 생략하였다.

예시)

로봇 수=4, 충전기 수=2, $Dist_{NewWork}$=12, 각 로봇의 초기 $Dist_{extra}$는 40,

한 로봇이 일을 끝내는 동안 충전소에 있는 로봇이 충전하는 Dist값=24

$\left(\left\lfloor\frac{i-1}{충전기\ 수}\right\rfloor+1\right)Dist_{NewWork}$에 i를 1부터 4까지 넣어 계산하면 아래와 같다.

12, 12, 24, 24

로봇의 $Dist_{extra}$들을 시간 흐름에 따라 관찰하면 아래와 같다. 왼쪽은 문제가 해결되지 않은 수식을 사용한 결과이고, 오른쪽은 개선된 수식의 결과이다.

1	2	3	4		일하는 로봇 수
40	40	40	40		4
28	28	28	28		4
16	16	16	16		4
4	4	4	4		0
6	6	4	4		0
5	5	6	6		0
7	7	5	5		0
6	6	7	7		0

1	2	3	4		일하는 로봇 수
40	40	40	40		4
28	28	28	28		2
16	16	52	52		2
4	4	76	76		2
28	28	64	64		2
52	52	52	52		2
76	76	40	40		2
100	100	28	28		2

충전중인 로봇은 주황색으로, 충전 대기중인 상태는 초록색으로 표시하였다. 오른쪽의 경우 왼쪽과 달리 미리 일하는 로봇의 수를 조절하여 일하는 로봇 수가 일정한 것을 확인 가능하다.

- 배터리 최소량 조정

배터리는 일반적으로 70% 내에서 사용하는 것이 좋다고 한다. 0% 가까이 배터리를 사용하면 수명이 급격히 저하한다. 배터리 최소량을 정해줄 필요성이 있다. 사실 위의 수식에는 빠진 부분이 있다.

$$\text{if}(Dist_{extra} >= B_{fail}\cdot\frac{V}{C}+\left\lfloor\frac{i-1}{충전기\ 수}\right\rfloor Dist_{NewWork}$$

B_{fail}은 로봇이 작동불능이 되는 배터리의 양, 즉 0%이다. 그런데 이 B_{fail}의 값을 70%로 설정하면 배터리는 70% 이상의 영역에서만 작동하게 된다.

- 업무 분배

업무 배열에는 처리되어야 하는 일들이 10개정도 쌓여 있다. 앞선 절들의 절차를 통해 로봇에게 일을 주어야겠다고 판단하면 로봇은 각 일들 중 어떤 일이 자신에게 가장 적합한지 BFS를 통해 알아낸다. 로봇은 동선이 가장 짧은 일을 택한다.

규모가 큰 실제의 물류공장이라면 변두리에 시작점이 있는 업무의 경우 로봇들에게 지속적으로 선택받지 못할 가능성이 존재한다. 이러한 경우 지속적으로 선택받지 못한 업무를 강제로 로봇에게 할당하는 방법 등이 있겠다. 그러나 우리의 관심사는 충전/업무부여의 여부이지 업무 분배가 아니므로 무시하도록 한다.

성능평가 및 마무리

이상 배터리 잔량에 따른 업무부여 알고리즘을 알아보았다. 현재 이 알고리즘을 C++ 콘솔창을 통해 시뮬레이팅하고 있다. 이 알고리즘의 성능을 평가하기 위한 방법에 대해 아래의 절에서 설명하도록 하겠다.

알고리즘 테스트 화면

- 알고리즘의 평가

평균적으로 일하는 로봇의 수와 평균적으로 처리되고 있는 일의 수, 시간당 처리되는 일의 표준편차, 목표 대수 미만으로 일한 시간의 비율 등의 값을 평가한다. 여러 다른 알고리즘을 만들어 두 값을 얻어낸 후, 본 알고리즘과 비교한다.

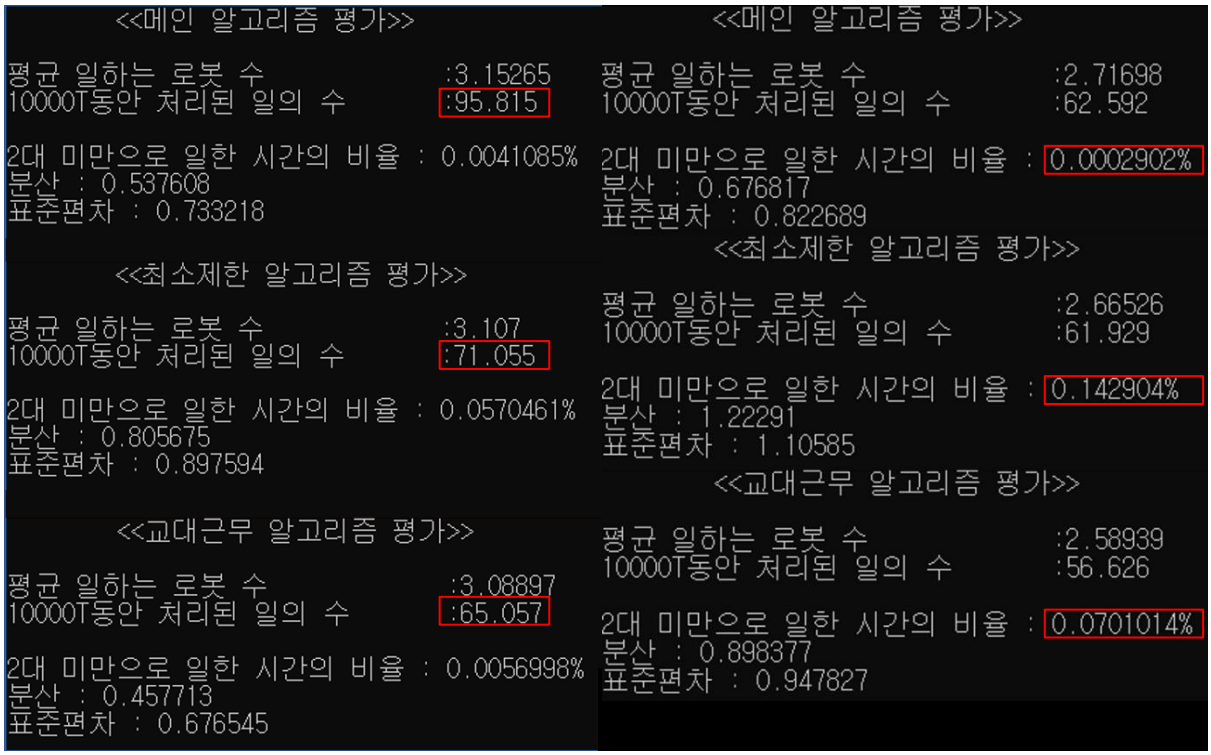
비교 알고리즘 A. 여러 로봇이 일정 주기로 번갈아서 충전.

비교 알고리즘 B. 항상 로봇들이 일을 하다가 배터리가 최소 제한량 아래로 내려가면 충전.

환경 설정은 다음과 같다.

로봇의 수는 4대, 각 로봇의 초기 배터리량은 (15, 30, 45, 70), 일을 한번 할 때 소모되는 배터리량은 동선에 따라 10~15, 충전기까지 가는 데 소모되는 배터리량은 동선에 따라 3~6의 랜덤값을 가진다. 최소 2대가 항상 일하는 것을 목표로 한다.

단위시간당 배터리 소모량은 충전량의 30%, 50%로 설정하여 비교한다.



왼쪽은 소모량이 충전량의 30%, 오른쪽은 소모량이 충전량의 50%

소모량이 상대적으로 적은 경우, 평균적으로 일하는 로봇의 수는 거의 차이가 없었다. **전체 시간 동안 처리된 일의 수**는 **메인**, 최소제한, 교대근무 알고리즘 순으로 **96**, 71, 65로 뚜렷한 차이를 확인할 수 있었다. 2대 미만으로 일한 시간의 비율은 최소제한 알고리즘의 경우가 0.057퍼센트로 가장 높았고 메인 알고리즘과 교대근무 알고리즘은 10배 작은 0.005퍼센트에 근접했다. 교대근무 알고리즘이 메인 알고리즘보다 표준편차가 더 낮았다는 점을 제외하면 모든 기준에서 메인 알고리즘이 다른 두 알고리즘에 비해 우수함을 확인할 수 있었다.

소모량이 상대적으로 많은 경우, 메인 알고리즘은 모든 기준에서 다른 두 알고리즘보다 우수한 수치를 보였다. 평균적으로 일하는 로봇 수는 가장 높았으며 전체 시간동안 처리된 일의 수도 가장 높았고 2대 미만으로 일한 시간의 비율은 200배 이상 낮았다. 표준편차 또한 가장 낮은 수치를 보였다.

- 결론

본 알고리즘은 현 시점에 실제 무인 물류공장의 AGV에 적용되는 다른 알고리즘에 비해 업무 처리량, 업무 처리량의 편차, 시간당 일하는 로봇의 수 등 많은 지표에서 우수한 성능을 보였다.