공학 연구논문집 제25권 제2호 pp. 227~234, 울산대학교
Journal of Engineering Research Vol. 25, No. 2, pp. 227~234, 1994.
University of Ulsan.

3-Trolley Portainer Crane 운전 시뮬레이터의 개발

최제헌 · 김규년 · 정민포 · 정태동* · 김종훈*
전자계산학과

<요 약>

항만에서, 빠른 시간내에 많은 양의 컨테이너 박스를 운반하는것이 곧 비용 절감이다. 이를 위해서는 크레인의 성능을 미리 예측을 할 수 있어야 한다. 특히 새로운 크레인을 제작할때, 실제 시험을 해본후 그 성능을 알아내기란 어려운 실정이다. 따라서, 컴퓨터상에서 이를 미리 시뮬레이션하여 그 성능을 알아낸다면 여러면에서 비용절감의 효과를 얻을수 있다.

본 연구에서는 기존의 하나의 Trolley에 의해 운전되는 크레인보다 더 나은 성능을 위해 설계된 3-Trolley Portainer Crane을 시뮬레이션하는 소프트웨어를 개발했다. 이것은 3개의 Trolley가 동시에 작동하여 하나일때보다 더 빠른 시간에 작업을 완수할수가 있다. 이 사실을 시뮬레이션을 통해 증명하고, 다양한 결과 자료를 이용해서 실제 이 크레인의 제작에도 상당히 도움이 될수 있다. 또한, 3개가 동시에 움직여서 각 구간별 소요시간의 측정이 문제가 되지만, 이것을 하나의 task로 미리 계산할 수 있는 방법을 고안, 적용하여 완수 하였다.

A Development of Operation Simulator to 3-Trolley Portainer Crane

Choi, Jae-Heon · Kim, Kyoo-Nyun · Jung, Min-Po
Jung, Tae-Dong* · Kim, Jong-Hun*
Dept. of Computer Science
*Industrial Plant Division, HYUNDAI Heavy IND. CO., LTD

<Abstract>

In the port, it is a reduction of cost to load or unload many containers quickly.

---

* 현대중공업 플랜트 사업부

To do so, we should predict the performance of crane. Therefore, if we could simulate a crane operation and estimate the performance by the computer system, we can obtain a benefit of cost.

In this study, we developed an operation simulator system of the 3-Trolley Portainer Crane which is designed for more efficeincy. Conceptually, the 3-Trolley Portainer Crane is operated concurrently by 3 trolleys, so it is faster than 1-Trolley Portainer Crane. We proved that through the simulation.

It will be useful to manufactrue 3-Trolley Portainer Crane with many results from this software system. Also, the problem that must be calculated in parallel is solved by a single calculation method which we developed.

1. 서 론

항만에서 컨테이너 박스를 배에 싣거나 육지로 내리는 역할을 하는 Portainer Crane (Port Container Crane)은 국제 무역량이 늘어남에 따라 점점 그 건조 기술 및 성능이 향상되어지고 있다. [1]

현재 대부분의 Portainer Crane은 [그림 1]과 같은 single trolley로 운영되고 있다. 그러나, 이러한 원리의 크레인은, 크레인에 사용되는 모터와 각종 설계기술에 의하여 단위시간당 옮길 수 있는 컨테이너 박스의 양은 제한되어 있어서, 보다 빠른 개념의 Portainer Crane의 필요성이 제기되었다.

이러한 속도문제를 부분적으로 해결할 수 있는 3- Trolley Portainer Crane의 개념은 이미 약 20년전에 발표되었으나, 실제적인 효과를 검증하는것은 간단하지가 않았으므로 미뤄져 왔다. 크레인 수주를 받을 경우, 건설하려는 크레인의 효율성을 정확히 예측하여 발주측에 알려주는 일이 경쟁이 치열한 수주업무에서 중요한 일이므로, 제작하려는 크레인의 성능을 시뮬레이션으로 예측하는 것은 중요한 작업중의 하나이다.

따라서, 본고에서는 Trolley를 3개 가지는 크레인의 성능을 예측할 수 있는 simulator의 설계 및 구현과정과 그 결과를 살펴보고자 한다.

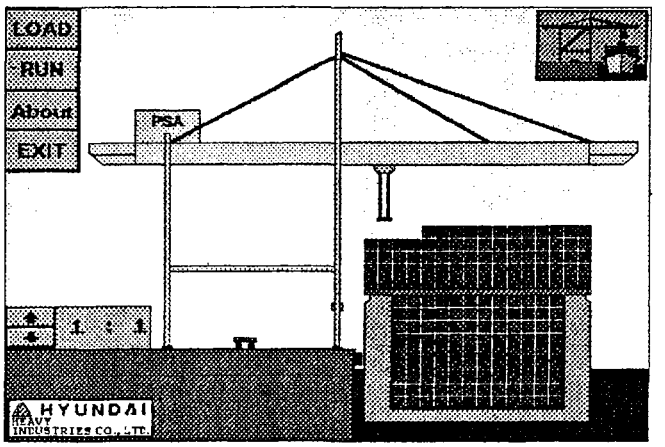
2. 본 론

2.1 3-Trolley Portainer Crane 의 개요.

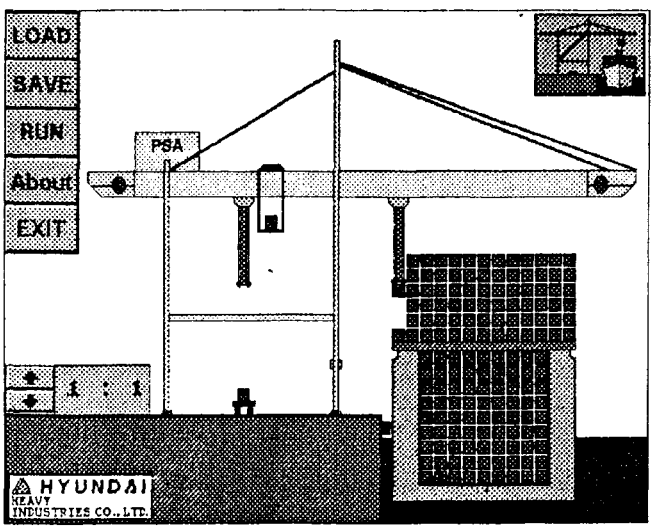
Trolley를 3개 가지는 크레인의 대체적인 형상은 [그림 2]와 같다. 그림에서 바다측과 육지측의 trolley는 컨테이너 박스를 들어올리거나 내리는 역할을 수행하고, 가운데 있는 trolley는 이들 사이를 움직이면서 컨테이너 박스를 전달받아 넘겨주는 역할을 담당한다.

예를 들어, [그림 2]의 경우에서, 바다측의 trolley는 배로부터 컨테이너 박스를 들어올리는 역할을 수행하고, 중간 trolley는 이를 전달받아 육지측 trolley에게 전달한다. 육지측 trolley는 좌우 이동은 없이 수직이동만 하므로써, 넘겨받은 박스를 하단의 차대에 전달하게 된다. 반면, 바다측의 trolley는 수직이동뿐만 아니라 수평이동도 하게 된다.

2.2 소프트웨어 개발시에 고려한 사항



[그림 1] Single Trolley Portainer Crane System



[그림 2] 3-Trolley Portainer Crane Simulator System

본 연구에서는 크레인이 주어진 환경의 컨테이너 박스를 모두 싣거나 내리는데 소요되는 시간과 각 박스당 소요되는 시간계산과 단위시간에 처리할 수 있는 컨테이너 박스의 갯수등을 계산할 수 있도록 해야 하며, 이와 아울러 다음과 같은 목표를 달성하려고 계획되었다.

① 이용하기 쉬운 사용자 인터페이스의 구현.

우선, 소프트웨어 시스템은 사용자가 쉽게 사용할 수 있는 방식으로 구현되어야 한다. 이를 위해서, [그림 2]와 같이 누구나 쉽게 알수 있는 메뉴 구성으로써, 이 소프트웨어를 쉽게 이용할 수 있도록 했다.

② 경과시간 계산의 구현.

바다측에서 배측으로, 배에서 바다측으로의 컨테이너 박스의 이동시에 여러가지 환경변수들을 설정하여, 실제로 소요되는 시

간을 각 구간 및 단위 컨테이너 박스마다 집계하고, 전체적인 통계확보를 위한 자료 정리를 수행한다.

③ 다양한 보고서의 출력.
한 척의 배전체를 대상으로 컨테이너 박스의 이동현황 및 이동시간을 그림 및 표형태로 제공하여 시스템 분석이 가능한 기초 자료를 제공한다.

2.3 입출력 사항 및 운용 환경

본 연구에서 개발된 소프트웨어 시스템의 구체적인 환경 및 기능은 다음과 같다.

① 시스템의 입력사항.
- -. 크레인 사양
- -. 컨테이너 박스의 사양
- -. 배의 사양
- -. 운전시의 변수들(slowdown, positioning, ...)
- -. 모터 관련 사양(속도, 구간 시간, ...)

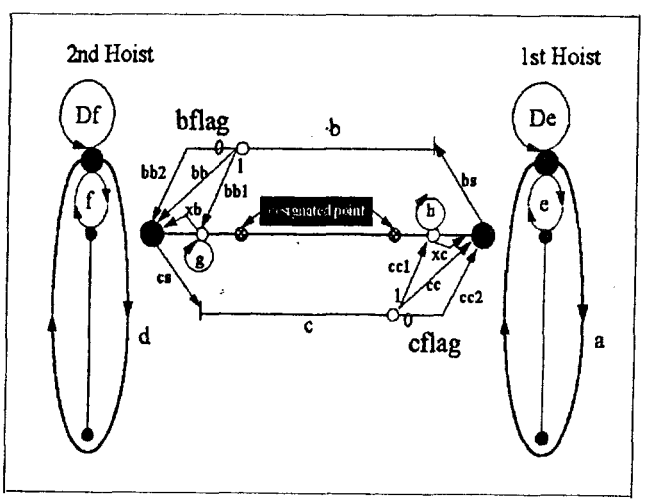
② 시스템의 출력.
- -. 각 컨테이너 박스의 구간별 이동 시간
- -. 전체 걸린 시간 및 각 컨테이너 박스당 소요 시간
- -. 단위 시간당 처리 박스 갯수

③ 시스템 사용 환경.
- -. IBM PC or 호환기종(80286, VGA이상)
- -. MS-DOS 5.0 이상
- -. EPSON 호환 프린터

2.3 3-Trolley의 multi-task를 single-task화

본 연구에서는 3개의 행위자가 움직이는 일련의 과정을 추적하여 이를 single task로 처리할수 있음을 발견하여, 이를 single task로 구현하였다. 이러한 이유는 MS-DOS환경하에서 multi-tasking[2]을 구현해야 한다는 어려움도 있었지만, 그 결과의 예측 불가능성(짧은 개발기간에 의한)을 사전에 배제하고자 한데에도 그 원인을 찾을수가 있다.

[그림 3]은 3개의 행위자의 움직임을 단일 task로 계산해 내기 위한 모델이며 loading path에서의 각 변수의 기능은 [그림 4]와 같다.



[그림 3] 각 Trolley및 구간별 운행 모델

[Loading path에서의 변수 정의]

d - 두번째 Hoist가 Top에서 출발하여, 육지의 컨테이너 박스를 들고 다시 Top에 돌아오는데 걸린 시간.

f - trolley가 안전거리내에 와서, 두번째 hoist 아래에 정지하는 순간부터 두번째 hoist가 slowdown으로 내려와서 컨테이너 박스를 trolley에게 전하고 다시 top까지 hoisting하는데 걸린 시간.

Df[n] - 두번째 hoist가 n번째 컨테이너 박스를 가진채로 top에서 정지하는 시간.

a[n] - 첫번째 hoist가 top에서부터 출발해서, n번째 컨테이너 박스의 위치를 찾아가서 내려놓고, 다시 top에 돌아오는데 걸리는 시간.

e - Trolley가 안전거리 안으로 들어와서, 첫번째 Hoist의 아래에서 정지하는 순간부터 첫번째 hoist가 slowdown으로 내려와서 컨테이너 박스를 Trolley로부터 전달받고 다시 top까지 hoisting하는데 걸리는 시간.

De[n] - 첫번째 hoist가 n번째 컨테이너 박스를 받기 위해 빈 상태로 top에서 정지하는 시간.

cs[n] - Trolley가 두번째 hoist의 아래에서 n번째 컨테이너 박스를 가진 상태로 출발해 안전거리를 벗어날때까지 걸리는 시간.

c[n] - Trolley가 n번째 컨테이너 박스를 갖고 두번째 hoist의 안전거리를 벗어나는 순간부터 첫번째 hoist의 designated point까지 가는데 걸리는 시간.

cc2[n] - Trolley가 n번째 컨테이너 박스를 갖고 첫번째 hoist의 designated point부터 행동을 계속해 바로 첫번째 hoist의 아래까지 들어가서 정지하는데 걸리는 시간.

cc1[n] - Trolley가 n번째 컨테이너 박스를 갖고 첫번째 hoist의 designated point부터 감속하여 안전거리에서 정지하는데 걸리는 시간.

h[n] - Trolley가 n번째 컨테이너 박스를 가진 상태로 안전거리에서 첫번째 hoist가 top에 도달하기를 기다리며 정지하는 시간.

xc - Trolley가 컨테이너 박스를 가진 상태로 안전거리에서 정지하고 있다가 첫번째 hoist가 top에 도달했을때 출발해서, 첫번째 hoist의 아래에서 정지할때까지 걸린 시간.

cc[n] - Trolley가 n번째 컨테이너 박스를 갖고 첫번째 hoist의 designated point부터 첫번째 hoist의 아래까지 가는데 걸리는 시간.

(cc2[n] 혹은 (cc1[n] + h[n] + xc)의 값을 가진다).

bs[n] - Trolley가 첫번째 hoist의 아래에서, n번째 컨테이너 박스를 전달한 후에 출발하여 안전거리를 벗어날때 까지의 시간.

b[n] - Trolley가 n번째 컨테이너 박스를 전달하고 돌아가는 도중에 첫번째 hoist의 안전거리를 벗어나는 순간부터 두번째 hoist의 designated point까지 가는데 걸린 시간.

bb2[n] - Trolley가 n번째 컨테이너 박스를 전달하고 돌아가는 도중에, 두번째 hoist의 designated point부터 두번째 hoist의 아래에 들어가서 정지하는데 걸린 시간.

bb1[n] - Trolley가 n번째 컨테이너 박스를 전달하고 돌아가는 도중에, 두번째 hoist의 designated point부터 속도를 감소시켜 안전거리에서 정지하는데 걸리는 시간.

g[n+1] - Trolley가 n+1번째 컨테이너 박스를 받기 위해 안전거리 밖에서 두번째 hoist가 top에 도달하기를 기다리며 정지하는 시간.

[그림 4] Loading path에서의 변수정의(continued)

xb - Trolley가 빈 상태로 안전거리 밖에서 정지하고 있다가 두번째 hoist가 top에 도달했을때부터 출발해서 두번째 hoist의 아래에서 정지할때까지 걸린 시간.
bb[n] - Trolley가 n번째 컨테이너 박스를 반납하고 돌아가는 중에 두번째 hoist의 designated point부터 두번째 hoist의 아래까지 가는데 걸린 시간.
(bb2[n] 혹은 (bb1[n] + g[n+1] + xb)의 값을 가진다)

[그림 4] Loading path에서의 변수정의(continued)

2.4 Crane 운전시간 계산 알고리즘

[그림 5]는 loading path시의 운전시간 계산 알고리즘을 보여주며, unloading path도 이와 유사하다.

```
/* 변수 초기화 부분 */
g[1] = d, Df[1] = xb, S = 0, h[1] = 0;
cc[1] = h[1] + cc2[1], De[1] = d+xb+f+cs[1]+c[1]+cc2[1];
for(n = 2; n <= N; n++){
  bflag = (d < (c[n-1]+cc[n-1]+e+bs[n-1]+b[n-1])) ? 0:1;
  g[n] = (bflag == 0) ? 0 :
       (d - (c[n-1]+cc[n-1]+e+bs[n-1]+b[n-1]+bb1[n-1]));
  if(g[n] < 0) g[n] = 0;
  bb[n-1] = (bflag == 0) ? bb2[n-1] : (bb1[n-1] + g[n] + xb);
  Df[n] = abs(d - (c[n-1]+cc[n-1]+e+bs[n-1]+b[n-1]+bb[n-1]));
  S += (De[n-1] + e + bs[n-1] + a[n-1]);
  cflag = (a[n-1] < (b[n-1]+bb[n-1]+f+cs[n]+c[n])) ? 0 : 1;
  h[n] = (cflag == 0) ? 0 :
        (a[n-1] - (b[n-1]+bb[n-1]+f+cs[n]+c[n]+cc1[n]));
  if(h[n] < 0) h[n] = 0;
  cc[n] = (cflag == 0) ? cc2[n] : (cc1[n]+h[n]+xc);
  De[n] = abs(a[n-1] - (b[n-1]+bb[n-1]+f+cs[n]+c[n]+cc[n]));
}
bb[N] = bb1[N];
Total = S + De[N] + e + bs[N] + max(a[N],b[N]+bb1[N]);
```

※ 다음은 하나의 박스가 가지는 데이타 구조이다.

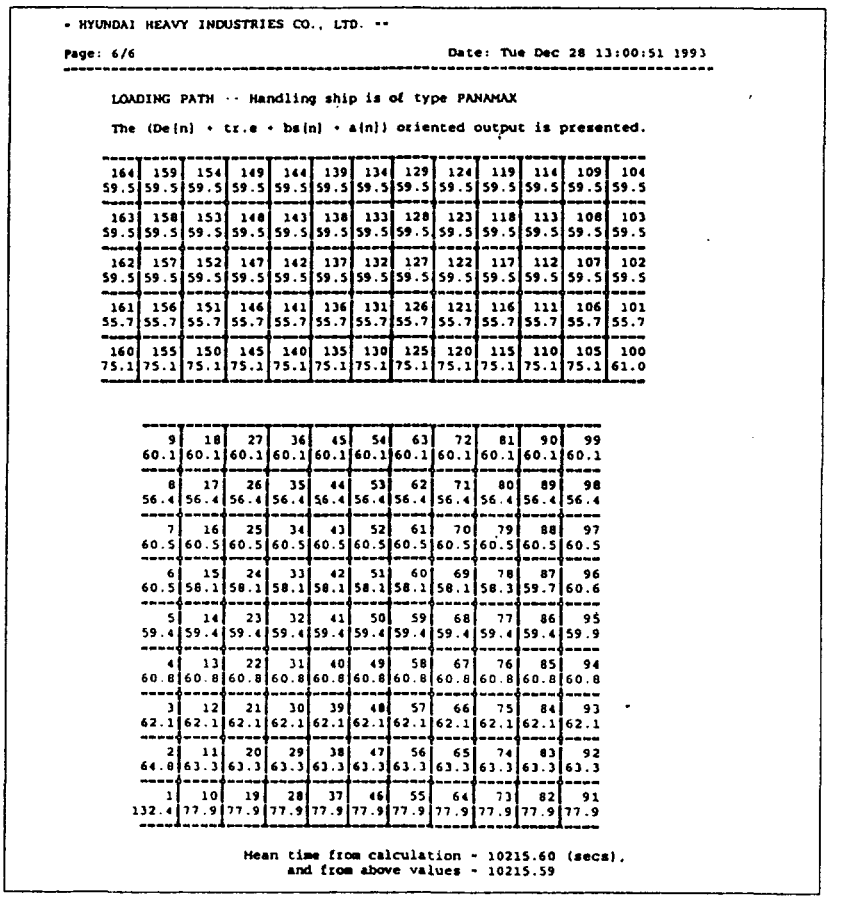
```
Box[n] = { a[n],  De[n], bs[n], b[n],   bbvalue, g[n], Df[n], cs[n], c[n],
           cc[n]-h[n],   h[n]  };

n = {1,2,...,N-1}, bbvalue = bb[n] - g[n+1];
n = N,             bbvalue = bb[n];
```

[그림 5] Loading path 운전시간 계산 알고리즘

2.5 시뮬레이션 결과

본 시스템의 시뮬레이션의 결과, single trolley와 직접적으로 비교하기에는 무리가 따르지만, 대체로 3-trolley crane의 효율이 single trolley crane보다 약 30%정도 향상될수 있음을 확인하였다.

[그림 6]에서는 PANAMAX형의 컨테이너선에 loading하는 경우의 각 박스의 이동시간을 계산한 결과중의 하나를 보여준다.



• HYUNDAI HEAVY INDUSTRIES CO., LTD. ••

Page: 6/6 Date: Tue Dec 28 13:00:51 1993

LOADING PATH •• Handling ship is of type PANAMAX

The (De[n] + tr.e + bs[n] + a[n]) oriented output is presented.

164	159	154	149	144	139	134	129	124	119	114	109	104
59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5
163	158	153	148	143	138	133	128	123	118	113	108	103
59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5
162	157	152	147	142	137	132	127	122	117	112	107	102
59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5
161	156	151	146	141	136	131	126	121	116	111	106	101
55.7	55.7	55.7	55.7	55.7	55.7	55.7	55.7	55.7	55.7	55.7	55.7	55.7
160	155	150	145	140	135	130	125	120	115	110	105	100
75.1	75.1	75.1	75.1	75.1	75.1	75.1	75.1	75.1	75.1	75.1	75.1	61.0

9	18	27	36	45	54	63	72	81	90	99
60.1	60.1	60.1	60.1	60.1	60.1	60.1	60.1	60.1	60.1	60.1
8	17	26	35	44	53	62	71	80	89	98
56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4	56.4
7	16	25	34	43	52	61	70	79	88	97
60.5	60.5	60.5	60.5	60.5	60.5	60.5	60.5	60.5	60.5	60.5
6	15	24	33	42	51	60	69	78	87	96
60.5	58.1	58.1	58.1	58.1	58.1	58.1	58.1	58.3	59.7	60.6
5	14	23	32	41	50	59	68	77	86	95
59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.9
4	13	22	31	40	49	58	67	76	85	94
60.8	60.8	60.8	60.8	60.8	60.8	60.8	60.8	60.8	60.8	60.8
3	12	21	30	39	48	57	66	75	84	93
62.1	62.1	62.1	62.1	62.1	62.1	62.1	62.1	62.1	62.1	62.1
2	11	20	29	38	47	56	65	74	83	92
64.8	63.3	63.3	63.3	63.3	63.3	63.3	63.3	63.3	63.3	63.3
1	10	19	28	37	46	55	64	73	82	91
132.4	77.9	77.9	77.9	77.9	77.9	77.9	77.9	77.9	77.9	77.9

Mean time from calculation = 10215.60 (secs).
and from above values = 10215.59

[그림 6] PANAMAX형의 컨테이너선에 loading하는 경우의 각 박스별 이동시간 계산 결과

3. 결론 및 앞으로의 방향

본 논문에서는 3-Trolley Portainer Crane의 운전시간을 시뮬레이션했다. 이를 위하여, multi-task를 single-task로 전환하는 모델을 개발하여 이를 근거로 구현하였음을 보여 주었다.

그 결과로 현재 널리 사용되는 single trolley crane보다 약 30%정도의 효율을 이룰수 있음을 확인하였다.

앞으로 이 시스템은 여러가지로 그 기능의 확장을 꾀할수가 있다. 예를들면, 첫째, 어떤 형태의 변수들의 값일때 최적의 효과가 나는지를 역으로 구할수 있고, 둘째, 또 다른 새로운 크레인의 설계시에 참고 가능한 시스템으로 확장이 가능할 것이다.

참고문헌

[1] "Lloyd's MaritimeAsia", Oct. 1992.

[2] Herbert Schildt, "Born To Code in C", Osborne McGraw-Hill, 1989.