Project Management

European Journal of Operational Research 274 (2019) 78–90

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

An efficient ant colony optimization algorithm for the blocks relocation problem

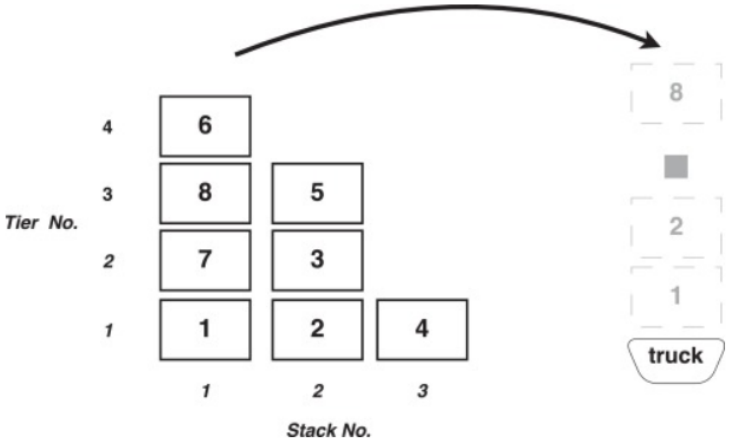
Raka Jovanovic[a], Milan Tuba[b], Stefan Voß[c,d,*]

[a] Qatar Environment and Energy Research Institute (QEERI), Hamad bin Khalifa University, Doha PO Box 5825, Qatar
[b] Graduate School of Computer Science, John Naisbitt University, Bulevar umetnosti 29, Belgrade, Serbia
[c] Institute of Information Systems, University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
[d] Escuela de Ingenieria Industrial, Pontificia Universidad Católica de Valparaíso, Chile

발표자: 이지호

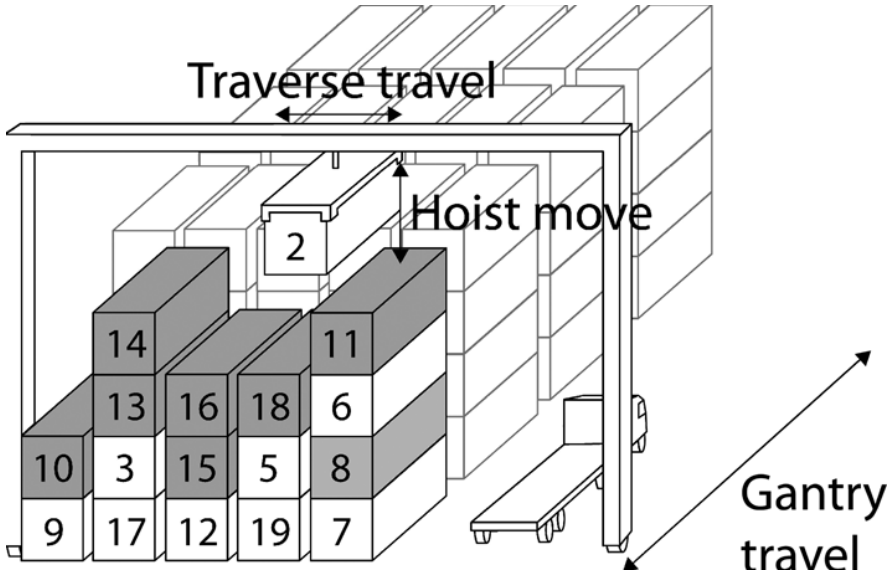
What is BRP (Blocks relocation problem)?

- 컨테이너의 due date를 기준으로 컨테이너를 재 배치하는 문제
  - ✓ 국제무역의 증가로 인해 항구에서의 효율적 컨테이너 적재가 더욱 요구됨
  - ✓ 항구의 에너지를 소비하는 주된 원인은 쌓기(stacking)와 수직이동(horizontal transport)임
  - ✓ BRP는 restrictedBRP (rBRP)와 unrestrictedBRP (uBRP) 두가지로 나뉨
    - rBRP는 well-located에 컨테이너를 옮길 수 없음
    - uBRP는 아무 공간에 컨테이너를 옮길 수 있음



Terminology

1. Stack: 컨테이너를 올릴 수 있는 공간
    - 본 논문에서는 s(c)로 표현
2. Tier: stack에 현재 쌓여 있는 컨테이너의 높이
    - 본 논문에서는 t(c)로 표현
3. Well-Located: due date of t© > due date of t(c+1)인 경우
4. Target: 가장 작은 due date를 갖는 컨테이너
5. top(t): target이 존재하는 stack의 가장 위에 있는 컨테이너



Definition of BRP

1. Bay는 2-dimensio이며 스택의 집합임 (각 스택은 최대 높이가 정해져 있음)
2. Bay에는 N개의 컨테이너가 존재하며, 각 컨테이너(c)는 1 to N의 due date를 가짐
3. Bay의 현재상태는 미리 알려져 있으며, s(c)/t(c)로 컨테이너의 위치를 나타냄
4. Stack의 가장 위에 있는 컨테이너만 이동 가능함
5. Bay에 남아있는 컨테이너 중, 가장 작은 due date를 가진 컨테이너를 최우선으로 빼내야 함
6. 컨테이너는 스택의 제일 위, 또는 빈 스택에만 올려놓을 수 있음
7. BRP의 목적은 컨테이너 재배치 회수를 최소화 시키는 것임

Greedy algorithm for the rBRP

```
while Bay not empty do
    Select target container t having the minimal due date
    while t not on top of stack do
        Select relocation stack for obstructing container c based on h
        Relocate c to selected stack
    end while
    Retrieve t
end while
```

주로 사용되는 heuristic function은 MinMax 알고리즘임

MinMax 알고리즘

- 알고리즘의 컨셉은 다음과 같음

  - ✓ 최소값의 due date를 갖는 컨테이너가 쌓여 있는 stack을 비워질 stack으로 선택
    - wasting slots (큰 due date를 갖는 컨테이너가 well-located가 되는 경우)을 최소화 할 수 있음
  - ✓ 비워질 stack을 제외한 stack 중 maximal value of the minimal due date of a container를 갖는 stack을 옮겨질 stack으로 선택
    - 이중작업 (옮긴 컨테이너를 다시 옮기는 일)을 최소화 할 수 있음

MinMax 알고리즘

$R_c = \{S \mid (S \in Stacks) \wedge (H_S < H) \wedge (S \neq s(c))\}$: all non-full stacks without c

$$dd(S) = \begin{cases} \min\limits_{c \in S} c & H_S > 0 \\ N+1 & H_S = 0 \end{cases}$$: nonempty/empty stack

$$dif(c,d) = \begin{cases} d-c & d > c \\ 2N+1-d & d < c \end{cases}$$: difference between c, d

$$dif(c,S) = dif(c, dd(S))$$

$$MinMax(c) = \arg\min_{S \in R_c} dif(c,S)$$

Basic greedy 알고리즘

$$\tilde{C} = \bigcup_{c \in Tops} (\{c\} \times R_c)$$

: all non-full stacks without c

$$W_c = \{S \mid S \in R_c \wedge dd(S) > c\}$$

: 이동대상 컨테이너보다 min값이 작은 stack

$$T_n = \{d \mid d \in Tops \wedge \neg WellLocated(d) \wedge s(d) \neq s(t)\}$$

: 잘 정돈되지 않았으며, target과 다른 컨테이너

$$T = \{top(s(t))\} \times R_{top(s(t))}$$

: 이동할 수 있는 후보집합

$$O_r = \bigcup_{c \in T_n} (\{c\} \times W_c)$$

: non- well-located 컨테이너 집합 (target 컨테이너 포함하는 스택 제외)

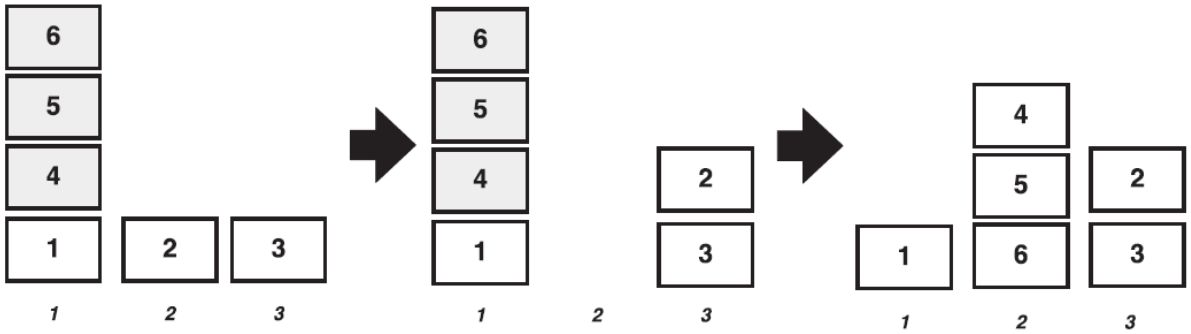
$$C_r = T \cup O_r$$

: T와 Or을 만족하는 모든 집합

$$MinMax(Bay) = \arg\min_{\alpha \in C_r} dif(\alpha)$$

Basic greedy 알고리즘

- Basic greedy 알고리즘은 well-located stack을 변경할 수 없다는 제약이 존재함



$T_n = \{d \mid d \in Tops \wedge \neg WellLocated(d) \wedge s(d) \neq s(t)\}$: 여기서 not_WellLocated(d) 빼면 안되나..

ACO approach

- Pheromone matrix

$\tau_{cdm_c t}$

c = 옮겨질 컨테이너
d = 목적지 stack (dd()함수로 계산된 dd로 표현)
m_c = c가 옮겨진 횟수
t = c가 왜 옮겨지는지 (target container)

- Transition rule

$$\alpha = (c, S)$$

$$f(c, d) = \frac{1}{1 + dif_e(c, d)}$$

$$f(c, S) = f(c, dd(S))$$

$$f(\alpha) = f(c, S)$$

$$g(\alpha) = f(\alpha)\tau_{cdm_c t}$$

$$select = \begin{cases} \arg\max_{\alpha \in C} g(\alpha) & q < q_0 \\ prob & q \geq q_0 \end{cases}$$

$$prob(\alpha) = \frac{g(\alpha)}{\sum_{\delta \in C} g(\delta)}$$

q0 is used to define the exploitation/exploration rate.
$q \in (0, 1)$ is a random variable

ACO approach

- Update rules

$$val(\mathcal{S}) = \frac{1}{|\mathcal{S}| - LB + 1}$$

|S| = Solution에서 발생한 relocation 횟수
LB = Solution의 Low bound
Val() => Solution 평가

Global Update rule

$$\Delta\tau = val(\mathcal{S}_{best})$$

Float

$$\tau_{cdm_c t} = (1-p)\tau_{cdm_c t} + p\Delta\tau \quad , \forall (c, d, m_c, t) \in \mathcal{S}_{best}$$

4-tuple

0/1

Local Update rule

$$\tau_{cdm_c t} = \varphi\tau_{cdm_c t} \, , \quad \forall (c, d, m_c, t) \in \mathcal{S}_i$$

0/1

Algorithm 2 Pseudo-code for the ACO algorithm for the BRP.

Generate solution $\mathcal{S}_g$ using the greedy algorithm
Calculate $LB(Bay)$
Initialize the pheromone matrix τ based on $\mathcal{S}_g$
while (Not Stopping Criteria) **do**
  for n ants **do**
    Clear Solution $\mathcal{S}$
    Reset array for tracking the number of relocations M
    Reset auxiliary structures
    while Bay not empty **do**
      Select Container target t having the minimal due date
      while t not on top of stack **do**
        Calculate candidate list C
        Select best relocation $\alpha \in C$ based on transition rule
        $\mathcal{S}.Add(\alpha.c, dd^*(\alpha.S), M[\alpha.c], t)$
        Apply relocation α to Bay
        Update auxiliary structures
        if $|\mathcal{S}| + LB(Bay) \geq |S_{best}|$ **then**
          $valid = false$
          break double while ▷ Stop generating current solution
        end if
      end while
    end while
    Apply local update rule for S
    Check if $\mathcal{S}$ is valid new best solution
  end for
  if $MaxConst$ iterations without improvement **then**
    Reinitialize pheromone matrix
  end if
  Apply global update rule for S_{best}
end while

감사합니다