Single Vehicle Routing in Port Container Terminals

Virginia Nunes Leal Franqueira

virginia.franqueira@terra.com.br

Advisers: Hannu Tapio Ahonen & Arlindo Gomes de Alvarenga

August/2003

Abstract

Export containers must be carried over from the port storage area to container ships to be delivered to their destination. Optimizing containers' transport routing is essential in order to enhance port performance and save costs. This thesis deals with a single vehicle routing problem in a container terminal environment. Heuristic strategies Beam Search and Ant Colony Optimization are proposed to solve the problem and are tested comparatively. A new strategy for container collection is proposed as a substitute for the traditional greedy strategy of container collection.

Contents

List of Algorithms

List of Figures

List of Tables

Acknowledgements

I must thank my advisers for their constant support during this long journey.

I must thank my parents for giving me life, love and principles.

I must finally thank my daughter and husband for "everything", including their daily dedication, patience, understanding and encouragement.

Chapter 1

Introduction

1.1 Problem context

The introduction of containers caused several changes in port environment and transportation by sea. It required innovations in the use of specialized equipments and revolutionized storage methods by taking advantage of containers' uniform shape which facilitates stacking.

There are two basic work flows for the container handling, depending on the final objective of the transport:

- import containers are unloaded from a ship, placed temporarily on a marshaling area, moved from there to storage areas and finally transferred to a terminal area for road or rail transportation to their destinations.

- export containers are unloaded from a truck or a train, placed into storage areas until their ship departure date and finally uploaded onto a ship heading to a determined location.

Therefore, port operations concerning containers essentially comprehend loading, stocking and transferring which have a direct impact on port service time and involves high costs. Port efficiency does not only involve service time but also waiting time which together define the berth time of the ships and affects their total traveling time and the transportation cost of the containers. Efficiency alone does not increase port customers' satisfaction since they also want to monitor container movements in real time systems, for example. All theses factors require optimization and improvements in port processes. Minimizing handling, transport and storage of containers becomes vital for both time and cost reasons.

This port scenario involves several computational problems, such as allocation of containers in port yards and also in ships, as well as routing and manipulation of containers. Kim and Kim [8] proposed a mathematical model for import container space allocation problem in container terminals. They focused on a segregation policy for stacking containers based on their arrival time, arrival rate and duration of stay. The objective was to minimize re-handling, i.e. prevent stacking containers which would stay longer to be placed on top. Kim *et al* [10] classified arriving export containers by weight, also considering their size and destination, to propose

a dynamic programming model for minimizing loading relocations. Ship stability is assured by placing heavier containers, which should be loaded first, below lighter containers, which should be loaded last. This principle guided their study which also allowed the extraction of decision trees in order to establish a load sequence list for load planners. Kozan and Preston [13] proposed an analytical scheduling model, and a solution based on genetic algorithm, in order to minimize handling for loading export containers into ships. Avriel *et al* [1] focused on the so called stowage planning, i.e. container storage allocation within a ship. The purpose of their work was to optimize ship storage space and minimize container shifting, which occurs when a container with an earlier departure date is stacked below a container with a later departure date. A Suspensory Heuristic procedure was proposed to resolve the problem. Preston and Kozan [16] proposed a solution to minimize container handling, i.e. placing and removing containers on storage areas, by optimizing involved operations using job scheduling. They suggested a solution based on Genetic Algorithm heuristic and discussed schedule versus random storage policies, changes to the yard layout and to the number of yard machines which perform loading. Kim and Kim [12] formulated the port routing problem for export containers during the loading process using Integer Programming. A Dynamic Programming solution was proposed to minimize the total travel distance of a single straddle carrier vehicle, used for container transportation between storage and marshaling areas. In [9] they proposed a Beam Search

procedure for resolving the same routing problem. An evaluation of algorithm performance was discussed based on numerical experimentation and comparison between Beam Search and Genetic Algorithm results.

1.2 The focus of the present work

Loading export containers on a ship requires three types of equipments: straddle carriers, yard trucks and quay cranes. Each one has a specific task in the overall loading process.

The straddle carrier has to move containers from where they are stored within a container terminal and to deliver them to a yard truck.

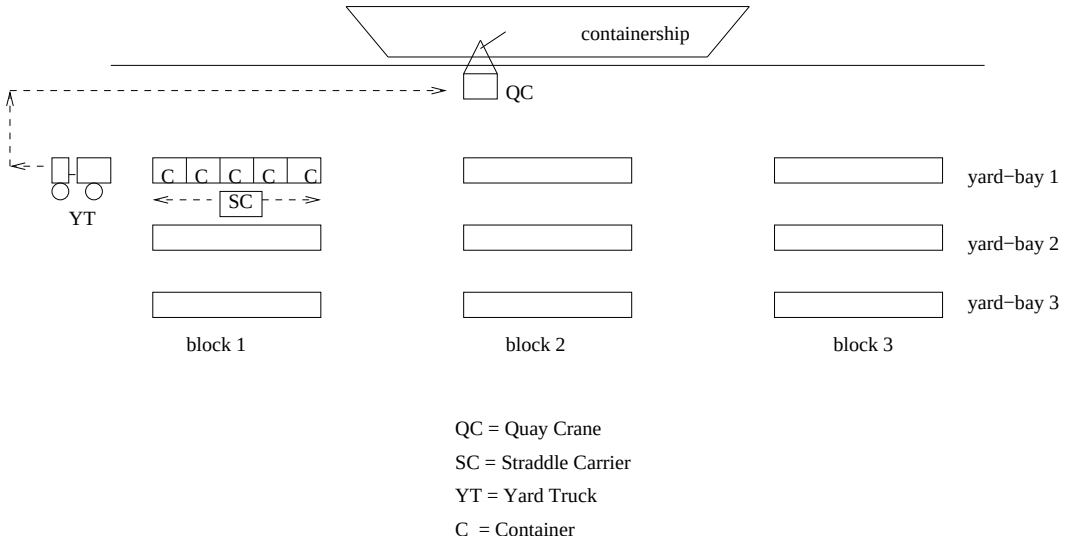
The yard truck, a combination of yard tractor and yard trailer, has to transport containers received from a straddle carrier to the marshaling area.

The quay crane has to pick up containers from the marshaling area and to place them inside container ships. It is a static equipment.

A container terminal yard is subdivided into blocks of yard-bays which contain containers arranged in rows. A *yard-map* shows the distances between blocks and between consecutive yard-bays.

Each straddle carrier, the only equipment allowed to enter yard-bays, is assigned to fulfill a quay crane loading sequence. This sequence defines a *work schedule* which determines the exact order in which containers must be handled

Figure 1.1: Port Container Terminal Diagram



and delivered by a straddle carrier (and consequently by a yard truck) to a quay crane.

Thus a straddle carrier has to accomplish the task of collecting all containers defined under the quay crane work schedule. In order to do that it has to move to a yard-bay which contains the specified container group, to pick-up the required quantity of containers and to move to the yard-bay end in order to be able to deliver all collected containers to a yard truck. The *container distribution table* gives the exact location of containers on a terminal yard.

The described scenario, Figure 1.1, involves two optimization problems which can be solved separately: determining a quay crane work schedule and determining a straddle carrier yard-bay visiting sequence. The first one is a transportation

problem and the goal is to find the exact order the quay crane must follow for ship loading. An optimal loading sequence as well as the right quantity to be loaded will reduce container ship loading time. The second problem is a vehicle routing problem and is highly affected by the work schedule determined by the first problem. The straddle carrier travel distance will be much longer if two consecutive containers must be collected far from one another instead of from consecutive yard-bays, for example. So, finding the minimum straddle carrier travel distance will guarantee port efficiency and allow cost savings.

The main focus of this work is the single straddle carrier routing problem in an export container terminal environment. No container stacking or weight constraints will be considered.

Chapter two describes and models the problem and discusses the problem complexity as a motivation for using heuristic solving methods, chapter three presents heuristic strategies Beam Search and Ant Colony Optimization, chapter four gives numeric results and compares both implemented strategies, chapter five discusses multiple vehicle routing problem and chapter six finalizes this thesis by presenting conclusions and pointing to future work.

Chapter 2

Single Straddle Carrier Routing Problem

2.1 Problem description

The objective of the straddle carrier (SC) routing problem is to minimize the total travel distance of a straddle carrier. It is bounded by the following restrictions and assumptions.

- a single SC is assigned to a single quay crane (QC);
- a SC must carry out a complete QC work schedule, following the exact order specified under the QC work schedule;
- container group repetition under the work schedule is allowed;

Table 2.1: A work schedule

Container group	CMB	ASH	HKG	SIN
Quantity	3	2	2	4

- all containers located under the container terminal must be collected by a SC;
- a SC is allowed to re-visit a yard-bay more than once;
- only distances between yard-bays will be considered since distances within a same yard-bay are constant.

The problem considers as provided:

1. the QC work schedule, which defines the work to be performed (Table 2.1);
2. the container stock or container distribution table, which defines the exact location of all containers at yard-bays (Table 2.2);
3. the container terminal yard-map, which defines distances between blocks and between yard-bays (Figure 2.1).

A SC performs a so called *partial tour* to pick-up containers of a same group, according to the work schedule. For example, in order to fulfill the first work

Figure 2.1: A yard-map

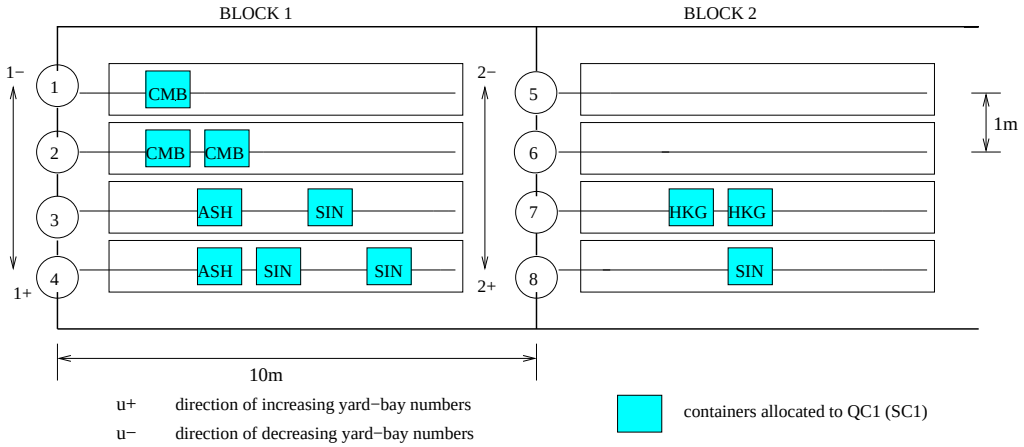


Table 2.2: A distribution of containers

block	yard-bay	ContainerGroup	Quantity
block-1	1	CMB	1
block-1	2	CMB	2
block-1	3	ASH	1
block-1	3	SIN	1
block-1	4	ASH	1
block-1	4	SIN	2
block-2	7	HKG	2
block-2	8	SIN	1

Table 2.3: Reviewed work schedule

Container group	CMB	CMB	ASH	HKG	SIN
Quantity	1	2	2	2	4

schedule item (CMB=3 from Table 2.1), the SC could move from block-1/yard-bay1 (pick-up one CMB) to block-1/yard-bay2 (pick-up two CMB). Therefore the first partial tour would be block-1/yard-bay1, block-1/yard-bay2.

Notice that a work schedule as in Table 2.3 with the first item as CMB=1 is perfectly possible. In this case however, the SC would stay at block-1/yard-bay1 because it only needs to pick-up one CMB to fulfill the first work schedule item. The first partial tour, in this case, would be block-1/yard-bay1. It is important to keep in mind that the work schedule must be accomplished exactly as stated and that, for each work schedule item, there is a partial tour associated which may define more than one SC movement.

Each partial tour item, composed of yard-bay, container group and quantity attributes, will define a so called *cluster* for implementations of this problem .

A *Tour* is a collection of partial tours, i.e. a tour defines all locations the SC shall visit in order to pick up all containers specified under the QC work schedule. Therefore the solution of the problem is a SC tour.

2.2 Problem formulation

The formulation of a single SC problem has been presented in [9].

Notation:

m = number of partial tours for a SC complete tour,

n = number of yard-bays,

l = number of container groups,

B = set of indexes of yard-bays $\{1, 2, \cdots, n\}$,

G = set of indexes of container groups $\{1, 2, \cdots, l\}$,

$S(h)$ = set of indexes of partial tours corresponding to container group h,

$B(h)$ = set of yard-bay numbers which contain containers of group h,

c_{hj} = initial number of containers of group h stacked at yard-bay j,

r_t = number of containers to pick up during partial tour t,

g_t = container group number to be picked up during partial tour t,

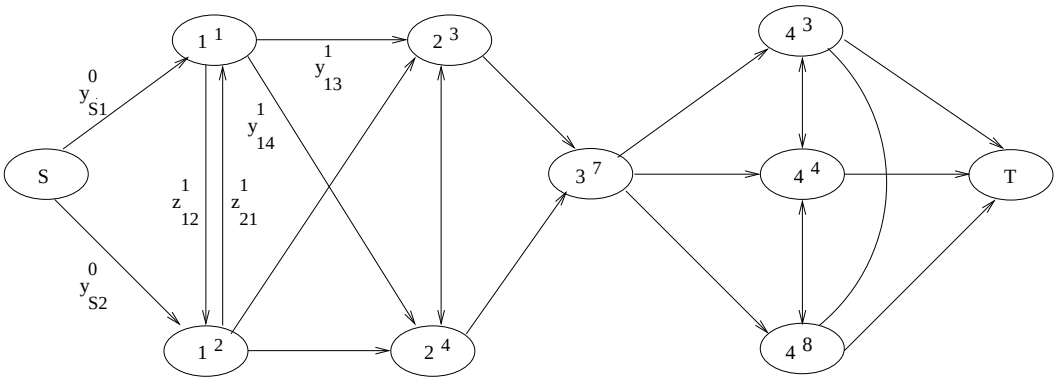
d_{ij} = travel distance between yard-bays i and j,

x_j^t = number of containers picked-up at yard-bay j during partial tour t

An SC tour can be expressed as a network graph represented by vertices (V) and arcs (A), where $A(V)$ is the set of arcs $A(V) = \{(i, j) \mid i, j \in V\}$, given a set of vertices V. For this problem, the vertice is represented as t^i where t is the partial tour number and i is the yard-bay number (Figure 2.2).

$B(g_0) = \{S\}$ source,

Figure 2.2: A network representation



$B(g_{n+1})$ ={T} sink,

Decision variables:

$y_{ij}^t = 1$, if SC moves from yard-bay i to j after completing partial-tour t; 0, otherwise

$z_{ij}^t = 1$, if SC moves from yard-bay i to j during a partial-tour t; 0, otherwise

The problem can be formulated as:

$$\min_{x_j^t,\, y_j^t,\, z_{ij}^t} \sum_{t=0,\, i \in B(g_t),\, \mathrm{j} \in \mathrm{B}(\mathrm{g}_{\mathrm{t+1}})}^{m} \sum d_{ij} y_{ij}^t + \sum_{t=1}^{m} \sum_{i,j \in B(g_t)} d_{ij} z_{ij}^t \tag{2.1}$$

Subject to:

$$\sum_{j \in B(g_t)} y_{Sj}^0 = 1 \tag{2.2}$$

$$-\sum_{j\in B(g_m)} y_{jT}^{m} = -1 \tag{2.3}$$

$$\sum_{j\in B(g_{t-1}),\, k\in B(g_t)} (y_{ji}^{t-1} + z_{ki}^{t}) - \sum_{j\in B(g_{t+1}),\, k\in B(g_t)} (y_{ij}^{t} + z_{ik}^{t}) = 0, i \in B(g_t),\ t = 1, 2, \cdots, m \tag{2.4}$$

$$\sum_{(i,j)\in B(g_t)} z_{ij}^{t} <\mid N \mid -1,\ \text{for all } N \subseteq B(g_t),\ t = 1, 2, \cdots, m \tag{2.5}$$

$$x_j^t \leq M(\sum_{k\in B(g_t)} z_{kj}^{t} + \sum_{i\in B(g_{t-1})} y_{ij}^{t-1}),\ j \in B(g_t),\ t = 1, 2, \cdots, m \tag{2.6}$$

$$\sum_{jEB(g_t)} x_j^t = r_t,\ t = 1, 2, \cdots, m \tag{2.7}$$

$$\sum_{t\in S(h)} x_j^t = c_{hj},\ j \in B(g_t),\ h = 1, 2, \cdots, l \tag{2.8}$$

$$y_{ij}^{t} \in 0, 1,\ i \in B(g_t),\ j \in B(g_{t+1}),\ t = 0, 1, \cdots, m \tag{2.9}$$

$$z_{ij}^{t} \in \{0,1\},\ i,\ j \in B(g_t),\ t = 1, 2, \cdots, m \tag{2.10}$$

$$x_j^t \geq 0,\ j \in B(g_t),\ t = 1, 2, ..., m \tag{2.11}$$

The first sum in Equation 2.1 represents the total distance traveled between partial-tours and the second represents the total distance traveled within a partial-

tour.

Constraints 2.2 to 2.4 guarantee flow conservation, defining/including the source node as a starting point and the sink node as a finishing point.

M in 2.6 is a large number.

Constraint 2.7 guarantees that containers picked up in a partial-tour correspond to the containers requested by the work schedule.

Constraint 2.8 guarantees that no containers will be left behind, i.e. containers stocked at yard-bays will all be picked up.

2.3 Problem complexity

First of all, some computer science concepts will be reviewed and then a study of the complexity of the Single Straddle Carrier Routing Problem (SSCRP) will be presented.

Computer devices (independent of technology limitations) can only solve problems for which an algorithm can be constructed, i.e. a sequence of instructions, leading to a guaranteed solution at the end. Alan Turing presented in 1936 an abstract model, called Turing Machine, for simulating what a physical computer is able to do. Therefore, he demonstrated there are problems which have an algorithm [6, 5], i.e. which can be solved by a halting Turing Machine. They are called decidable problems. However there are other problems which do not have

an algorithm, i.e. they never come to an end no matter how long they run on a halting Turing Machine. They are called undecidable problems.

A deterministic Turing Machine [14] is defined by $M = (K, \sum, \delta, s)$ where K is a finite set of states, $\sum$ is a finite set of symbols (or alphabet) which always contains "blank"($\sqcup$) and "start"($\triangleright$) symbols, δ is a transition function which maps $K \times \sum$ to $(K \cup \{h = haltState, "yes" = acceptState, "no" = rejectState\}) \times \sum \times \{left, right, stay\}$and $s \in K$ is the initial state. The machine initializes with state s, string symbol $\triangleright$ followed by a finitely long input string x, where $x \in (\sum - \{\sqcup\})$. From there, the machine performs a called action according to δ and determined by the state/symbol combination. Therefore it changes its state, prints a symbol and moves the cursor until it reaches one of the states h, yes or no. A nondeterministic Turing Machine follows the same concept as the deterministic one except by the fact there are choices of next actions, i.e. for each state/symbol combination there may exist more than one appropriate next step or even no step at all.

Algorithm complexity studies [14, 5, 6], although a vast theory field which involve many computational aspects, also deal with estimation of the time an algorithm takes to find a problem solution for each possible problem input. As such it determines how efficient an algorithm is. Computer scientists identify two important classes of algorithms complexity: polynomial time algorithms and exponential time algorithms. The first class comprehends algorithms for which the

Table 2.4: Comparison of polynomial and exponential time complexity functions

time complexity	size n				
function	10	20	30	40	50
n	0.00001 sec	0.00002 sec	0.00003 sec	0.00004 sec	0.00005 sec
n^2	0.0001 sec	0.0004 sec	0.0009 sec	0.0016 sec	0.0025 sec
n^3	0.001 sec	0.008 sec	0.027 sec	0.064 sec	0.125 sec
2^n	0.001 sec	1.0 sec	17.9 min	12.7 days	35.7 years
3^n	0.059 sec	58 min	6.5 years	3855 centuries	2x10^8 centuries

solution response time is bounded by a polynomial curve function for the problem input length (n). Therefore, algorithms with complexity $O(n^k)$ for $n > k \geq 1$ and also polynomial bounded algorithms with complexity $O(n \log n)$ are all considered as polynomial algorithms. The second class includes all algorithms with a non-polynomial response time, such as algorithms with complexity $O(k^n)$ for $k > 1$, $O(n^n)$, $O(n!)$ and $O(n^{\log n})$. Polynomial time algorithms usually resolve problems within an acceptable amount of time, while this is true to exponential time algorithms only in case of small instances of problems. If a problem cannot be resolved in polynomial time, it is called an intractable problem. According to [6], if a problem can be solved in polynomial time by a Turing Machine, it can also be solved in polynomial time by an ordinary computer and vice-versa. Table 2.4, extracted from [6], allows the comparison between polynomial and exponential time complexity functions.

According to [6], decidable problems can be classified as:

1. Class P (Polynomial): a problem belongs to this class if it can be resolved in polynomial time by a deterministic Turing Machine (TM);

2. Class NP (Nondeterministic Polynomial): a problem belongs to this class if it can be resolved in polynomial time by a nondeterministic Turing Machine. Since deterministic TM (without choices) is a special case of nondeterministic TM (with choices), P$\subseteq$NP.

3. Class NP-complete: a problem belongs to this class if it is a NP problem and any NP problem reduces to it in polynomial time.

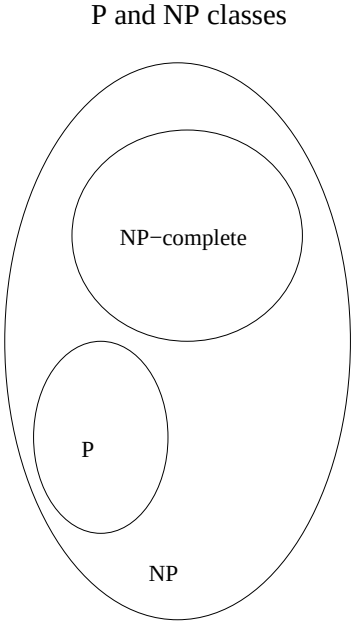
4. Class NP-hard: a problem belongs to this class if any NP problem reduces to it in polynomial time but there is no proof it is a NP problem, i.e. it is not certain it can be solved in polynomial time by a nondeterministic TM. A NP-hard problem is an intractable problem.

Note that whether P = NP is mathematical open question, however it is strongly believed the answer is negative [6]. If so, then P$\neq$NP, as on Figure 2.3.

Turing Reducibility [5, 7] is the ability to have an algorithm for converting instances of problem P_1 to instances of problem P_2 with the same answers. In this case, it is usual to say that P_1 reduces to P_2.

Polynomial-Time Reductions is a subset of the Turing Reductions [6, 15] which allows proving that, if there is a polynomial time reduction algorithm which

Figure 2.3: Class complexity (if P≠NP)



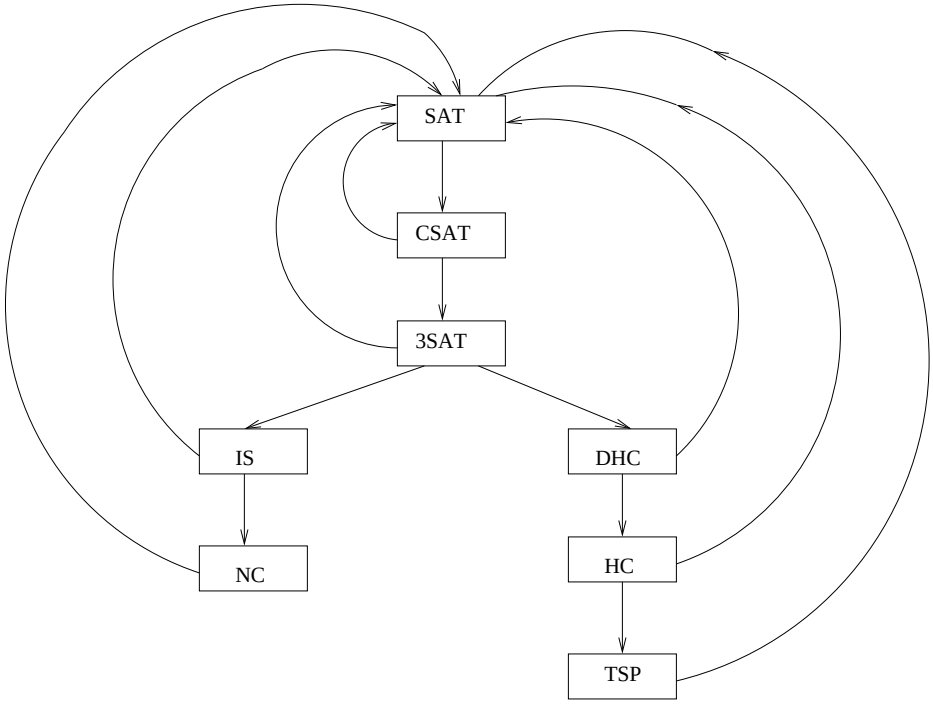
reduces a problem P_1 to a problem P_2 then, if P_1 does not belong to class P, P_2 will not belong either.

Figure 2.4, extracted from [6], shows reductions between NP-complete problems.

By theorem 10.24 from [6], we know that TSP (Traveling Salesman Problem) is a NP-complete problem.

Given a graph $G = (V, E)$ with n cities $(c_1, c_2, \cdots, c_n) \in V$ and numbers $d_{ij} > 0$, representing distance between cities, for each $[c_i, c_j] \in E$. The objective is to visit all cities and return to the initial city making a tour of the shortest length. This is the TSP problem.

Figure 2.4: Reductions among NP-complete problems



SAT − Satisfiability Problem
CSAT − Satisfiability for formula in CNF Problem
3SAT − 3_Satisfiability Problem
IS − Problem of Independent Sets
NC − Node−Cover Problem
DHC − Directed Hamilton−Circuit Problem
HC − Undirected Hamilton−Circuit Problem
TSP − Traveling Salesman Problem

Figure 2.5: Work schedule for the generic SSCRP problem

Container Group	cg_1	cg_2	$\cdots$	cg_n	$\cdots$	cg_1	cg_2	$\cdots$	cg_n
Quantity	1	1		1	$\cdots$	1	1		1
	—- 1 —-				$\cdots$	—- n —-			

Figure 2.6: Distribution of containers for the generic SSCRP problem

Container Group	cg_1	cg_2	$\cdots$	cg_n
yb_1	1	1		1
yb_2	1	1		1
$\cdots$				
yb_n	1	1		1

Now given the TSP problem (our problem P_1), we want to model a generic SSCRP problem (our problem P_2) such that the solution for the latter provides a solution for the former.

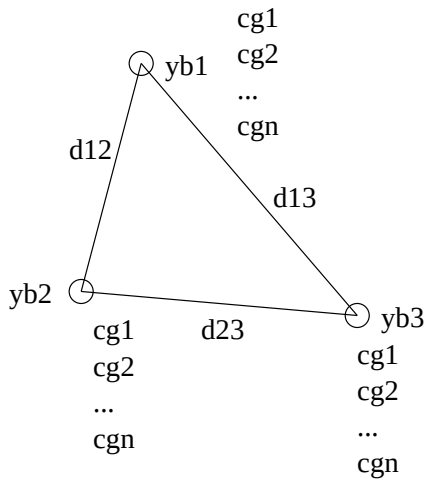
Suppose we have a work schedule as in Table 2.5 and a distribution of containers in yard-bays as in Table 2.6.

Each yard-bay belongs to a block and distance between blocks is defined as $d_{ij} > 0$.

If we consider the yard-bays (SSCRP) as cities (TSP) and the distances between blocks or yard-bays (SSCRP) as the distances between the cities (TSP), we can visualize a TSP to SSCRP conversion as in Figure 2.7.

In order to fulfill the generic SSCRP work schedule, the SC can follow two basic approaches:

Figure 2.7: The generic TSP problem mapped to a SSCRP problem



- collect one container belonging to container group 1 in yard-bay 1, then move to yard-bay 2 to collect one container belonging to container group 2 and so on performing n loops;

- collect one container of each group at yard-bay 1, then move to yard-bay 2 to collect one container of each group again and so on performing a single loop.

It is not difficult to see that a tour of shortest length would be obtained with the second choice, which corresponds to a solution for the TSP. Therefore a solution for the SSCRP is also a solution for TSP what means that, if a polynomial solution could be found for SSCRP (problem P_2), a polynomial solution for TSP (problem P_1) could be found as well. As no polynomial solution has ever been found for TSP [5], the same is true for SSCRP.

An interesting extension to this work would consist of trying to propose a non-deterministic TM algorithm for the SCCRP problem. In case such algorithm exists, SCCRP is NP-complete; if not, SCCRP is NP-hard. Considering that any NP problem can be reduced to TSP, according to Figure 2.4, and that TSP can also be reduced to SCCRP, as shown above, any NP problem could be reduced to SCCRP.

Since no polynomial-time solution for SSCRP can be found, exact algorithms can become time prohibitive (see Table 2.4) and heuristics are a good alternative. Heuristics [2] are methods which generally produce good solutions without any guarantee that an optimal solution will be found. Typically, two kinds of heuristic methods are used. The first kind attempts to construct the solution from scratch (called constructive heuristics) and the second kind attempts to improve an existing solution. This work will compare results obtained by a constructive heuristic method, Beam Search, with those obtained by an improving method, Ant Colony Optimization.

Chapter 3

Heuristic Strategies

3.1 Beam Search

Beam Search is a constructive heuristic method, used for solving large optimization and combinatorial problems. The method is based on the exploration of a search tree.

Each tree node generates a number of child nodes, however, only a specified number of these is considered for the next generation. This number is defined by a parameter called beam width. All alternatives are explored within this width and each path of the tree, from root to leaf, defines a complete solution for the problem (Figure 3.1).

A Beam Search tree for the example given on chapter 2 is represented in Figure 3.2.

Figure 3.1: Generic Beam Search Tree

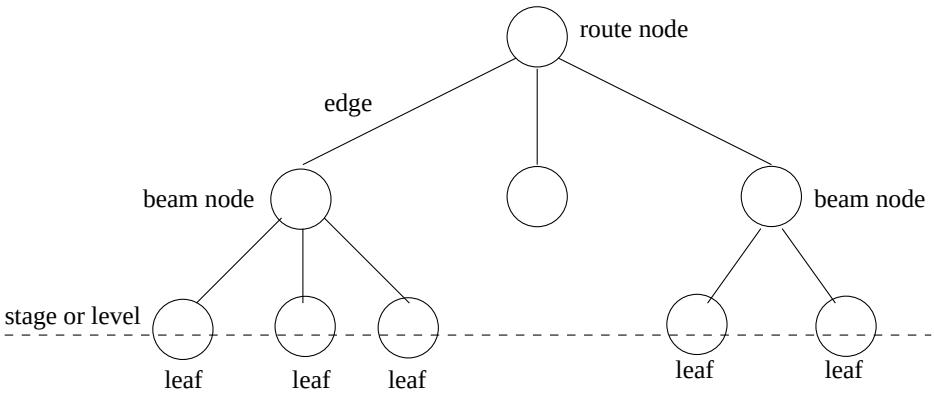
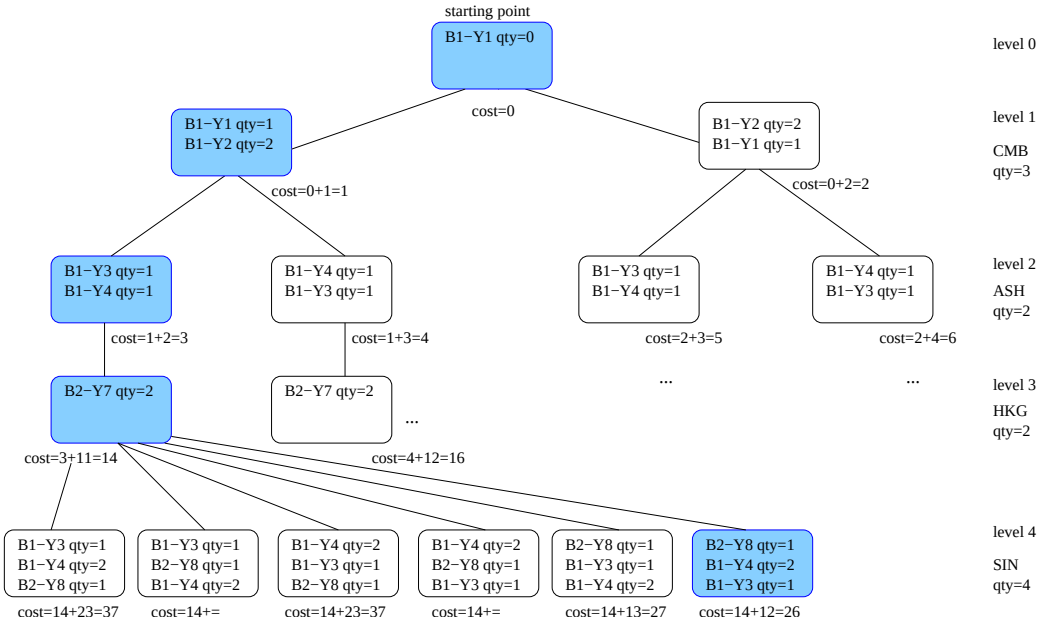


Figure 3.2: An example of a Beam Search Tree (width=2)



3.1.1 Details of Beam Search

3.1.1.1 High level heuristic algorithm

The high level Beam Search heuristic (Algorithm 1) builds the search tree by using the intuitive concept of father beam node generating child nodes. The specific structure *State*, which has father, level and cost attributes, is used in the formulation of the algorithm.

3.1.1.2 Implementation Notes

Method *addNextChildren* in Algorithm 1 is responsible for generating all possible child nodes from each beam tree father node. All yard-bays containing the container group being processed are included on a separate tree and, by covering this tree in-depth, we obtain every possible route. One child is generated for each route.

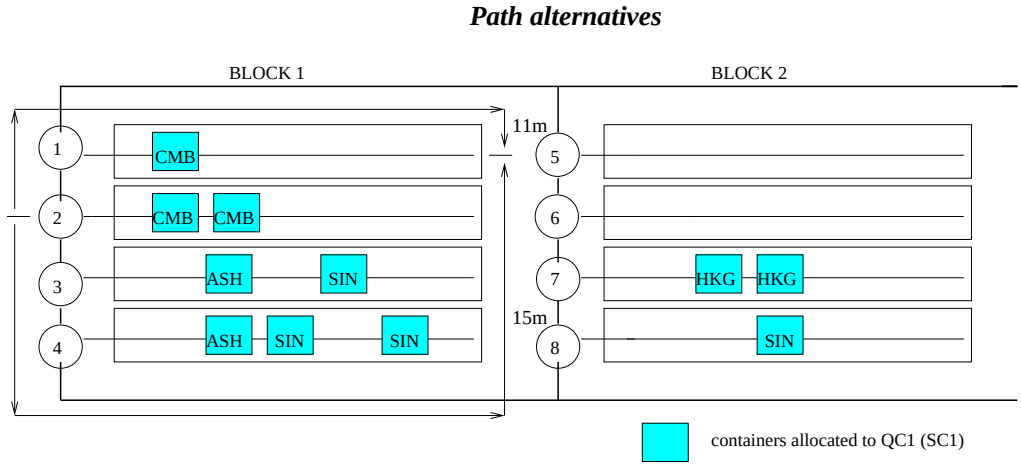
In the Beam Search heuristic the distance between two container locations, determined by block and yard-bay, is defined as the shortest linear path between the locations (Figure 3.3). Therefore the distance, or cost, to move a straddle carrier from block-1/yard-bay2 to block-2/yard-bay1 is 11m, considering that the distance between adjacent blocks is 10m and between adjacent yard-bays is 1m.

A solution is considered complete after the last work schedule item is committed, what means when the beam tree level reaches the work schedule size. Method

Algorithm 1 Beam Search High Level Algorithm

```
public State doBeamSearch (Problem problem, int BeamWidth) {

  State currentState = null;
  State solutionState = null;
  Vector openList = new Vector();
  Vector bestList = new Vector();
  Vector bextChildren = new Vector();

  openList.addElement(problem.getFirstState());

  while (openList.size() > 0) {
    bestList = selectBestStates(openList,beamWidth);
    openList.removeAllElements();

    while (bestList.size() > 0) {
      currentState = (State) bestList.firstElement();
      if (currentState.isSolutionComplete(problem)) {
        solutionState = getTheBestState(bestList);
        bestList.removeAllElements();
      }
      else {
         childrenList.removeAllElements();
         childrenList = currentState.addNextChildren(problem);
         for (int i=0; i<childrenList.size(); i++)
            openList.addElement(childrenList.elementAt(i));
         bestList.removeElement(currentState);
      }
    }
  }
  return solutionState;
}
```

Figure 3.3: Distance calculation from yard-bay 2 to yard-bay 5



isSolutionComplete in Algorithm 1 is responsible for verifying whether the beam search can be halted or not.

The list of the unexplored nodes (the "openlist") is ordered using a Bubble Sort procedure and the beam width best candidates in terms of cost are selected from this ordered list. Method *selectBestStates* in Algorithm 1 returns a list containing the selected nodes.

3.2 Ant Colony Optimization

Ant Colony Optimization is a heuristic method which simulates the behavior of natural ants finding their path from nest to food.

In nature, ants are able to find the shorter path between nest and food even when an obstacle exists on their way, as represented in Figure 3.4. They release

a certain constant amount of a substance called pheromone while moving along the way. Pheromone attracts other ants because they have a preference for paths with a large level of this substance accumulated. Therefore when their path is blocked, ants have to find their way out by getting around the obstacle. In this situation, it is expected that half of ants will choose one direction and the other half of ants will choose the opposite direction. Along the shortest way alternative, pheromone will accumulate quicker because ants, at a constant movement rate, will reach their food goal first and start their way back before than ants along the longest way alternative.

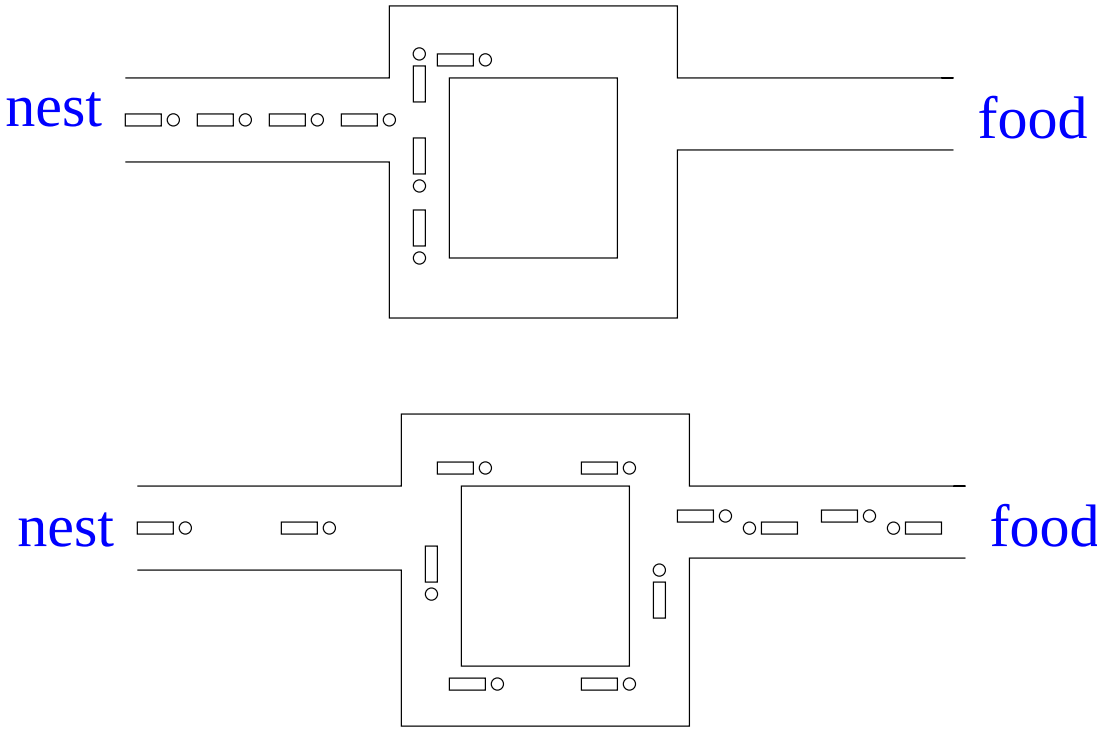
Ants' capacity to find the shortest way has been translated into artificial ant colony systems [3, 4] applied to optimization in real life problems. Artificial ants inherited two basic concepts from natural ants' behavior:

1. an amount of pheromone is deposited by each ant along its visited edges (defined by two points), what is is called Local Pheromone Update, according to Formula 3.1;

2. the shortest ant path is granted with a differential extra amount of pheromone, called Global Pheromone Update, according to Formula 3.2.

$$pheromone(edge) = (1 - \rho) * pheromone(edge) + \rho * \tau_0 \qquad (3.1)$$

Figure 3.4: Representation of ants in nature



$$pheromone(bestTourEdge) = (1-\alpha) * pheromeone(bestTourEdge) + \alpha * (\frac{1}{bestTourLength}) \quad (3.2)$$

In Formulas 3.1 and 3.2, ρ represents an evaporation factor; τ_0 represents a constant heuristic value and α represents a pheromone decay parameter value.

3.2.1 Details of Ant Colony Optimization

3.2.1.1 High level heuristic algorithm

The high level Ant Colony heuristic Algorithm 2 considers that each ant builds its solution during each iteration. The structure *AntSolution* in the algorithm plays a similar role as the structure *State* in Beam Search algorithm.

3.2.1.2 Implementation notes

The Ant Colony heuristic has a probabilistic parameter q for choosing between exploration or exploitation methods when determining an ant next location. If the current parameter value $q \in [0, 1]$ is less than or equal to the given $q_0 \in [0, 1]$ then the exploration procedure will be used, otherwise exploitation will be used.

Under exploitation, the candidate location with higher probability is chosen. The probability is calculated as in Formula 3.3.

Algorithm 2 Ant Colony High Level Algorithm

```
public AntSolution doAntSearch () {

  int iterationCount = 1;
  int bestCost = Integer.MAX_VALUE;
  int antCost;
  AntSolution antSolution;

  while (iterationCount <= maxIterations) {

    for (int i=1; i<=nAnts; i++) {
      antSolution = ant.buildSolution(problem,alpha,beta,q0,tau0,ro);
      antCost = antSolution.getCostValue();
      ant.printMe(iterationCount,i,antSolution,antCost,bestCost);
      if (antCost <= bestCost) {
        bestSolution = antSolution.updateBestSolution();
        bestCost = antCost;
        bestSolutionAnt = ant;
      }
    }
    problem.globalTrailUpdating(bestSolution,alpha,tau0);
    iterationCount ++;
  }
  return bestSolution;
}
```

$$probability = edgePheromone * (1/distance(currentLocation, candidateLocation))^{\beta} \quad (3.3)$$

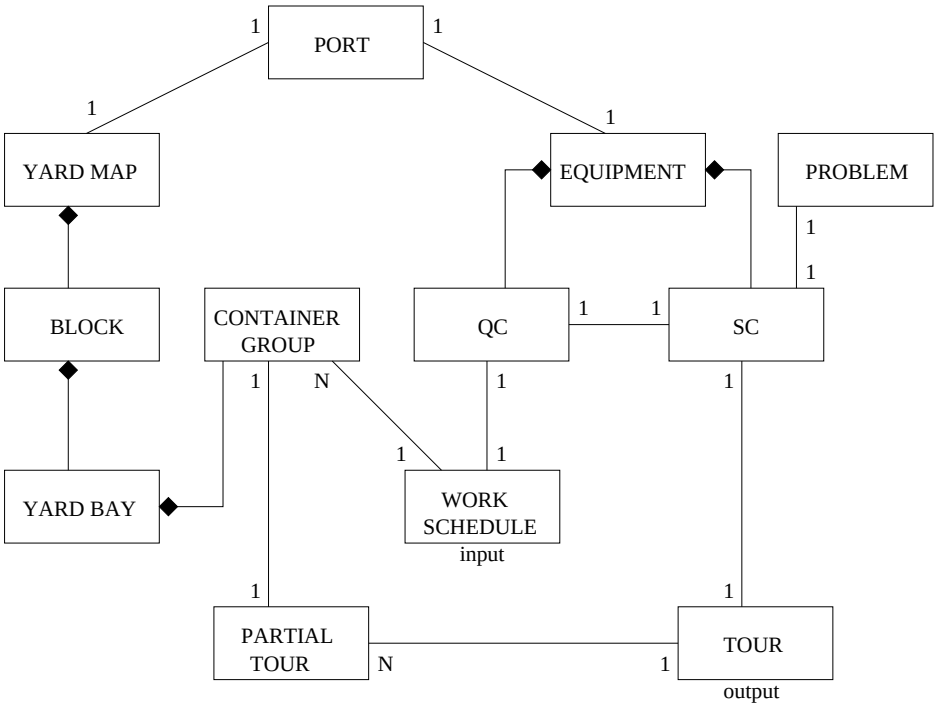
Under exploration, a roulette wheel is constructed with one slot for each candidate and with its size determined by probability as in Formula 3.3.

Ants build their own complete solution by following the given work schedule. Ants are initially placed on the starting point and then move to a next location determined by either exploration or exploitation methods. The quantity of containers collected at each location visited by the ant depends on the collection strategy adopted which can be the Greedy or the Random collection strategy, described in subsection 3.3.1. Method *buildSolution* in Algorithm 2 returns a complete ant solution.

Every time an ant moves from location A to location B, pheromone is deposited on the edge AB, according to Formula 3.1.

The best solution encountered so far is updated each time an ant builds a solution with cost smaller than the current best cost. After each iteration the best ant solution receives an extra amount of pheromone along its edges, according to Formula 3.2. Method *updateBestSolution* in Algorithm 2 is responsible for this update.

Figure 3.5: High level SSCRP Model



3.3 Solving the Routing Problem

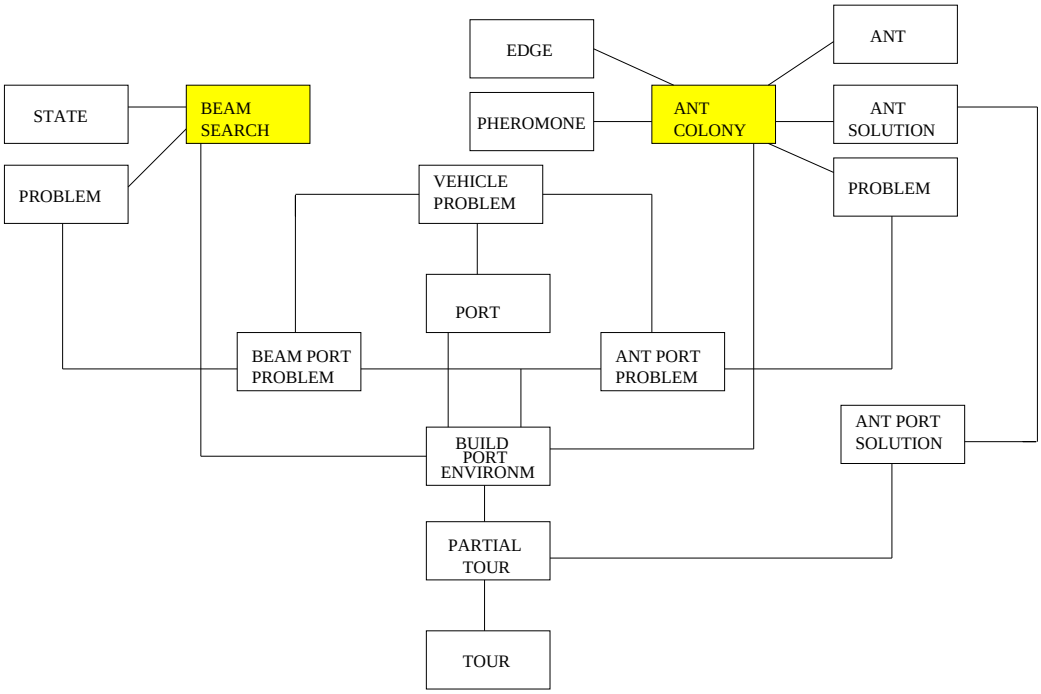
The port environment for the single routing problem has been modeled as depicted in Figure 3.5.

Figure 3.6 shows the approach chosen to resolve the single SC routing problem using Beam Search and Ant Colony strategies.

The principal elements of the model include:

- The class Port contains port environment settings;
- The class Vehicle Problem forms an interface between the problem and the

Figure 3.6: The model of SSCRP strategies



port environment classes;

- The class Build Port Environment uses the port settings, the problem itself and the respective strategy in order to provide a tour solution, which is nothing more than a chain of partial tours.

3.3.1 Container collection strategies

Two strategies have been developed in this thesis for collecting containers at the yard bays.

The first one refers to a greedy approach (Algorithm 3) and the second one refers to a random approach (Algorithm 4) for container collection.

The variables *stockQuant* and *reqQuant* (Algorithms 3 and 4) refer to the stored and required quantities of containers, respectively.

Algorithm 3 Greedy collection algorithm

```
if (stockQuant>0) {
  if (reqQuant>=stockQuant){
     // gets everything available
     clusters.addElement(new Cluster(yBay,containerG,stockQuant));
     collectQuant += stockQuant;
     reqQuant -= stockQuant;
  }
  else {
    // gets everything needed
    clusters.addElement(new Cluster(yBay,containerG,reqQuant));
    collectQuant += reqQuant;
    reqQuant = 0;
  }
}
```

The variables *randQuant* and *totChoice* (Algorithm 4) refer to a randomly generated quantity of containers and to the total quantity still stored at yard-bays, respectively.

The Greedy collection strategy collects the maximum quantity of containers available at a determined location.

The Random collection strategy collects a random quantity of containers with-

Algorithm 4 Random collection algorithm

```
if (stockQuant>0) {
  if (reqQuant>=stockQuant){
     randQuant=0;
     while(randQuant==0 || (reqQuant-randQuant-totChoice)>0)
        randQuant = (int)(Math.random()*1.2*stockQuant);
     if (randQuant>=stockQuant) {
       // gets everything available
       clusters.addElement(new Cluster(yBay,containerG,stockQuant));
       collectQuant += stockQuant;
       reqQuant -= stockQuant;
     }
     else {
       // gets random quantity
       clusters.addElement(new Cluster(yBay,containerG,randQuant));
       collectQuant += randQuant;
       reqQuant -= randQuant;
     }
  }
  else {
    // gets everything needed
    clusters.addElement(new Cluster(yBay,containerG,reqQuant));
    collectQuant += reqQuant;
    reqQuant = 0;
  }
}
```

out leaving behind containers needed to complete a work schedule partial tour. For example, suppose the work schedule first item demands that 10 containers of group A should be picked up. In yard-bay 1 you have 7 containers of group A and in yard-bay 2 you have 5 containers of group A. In this case, only random numbers $r \in \{5, 6, 7\}$ can be selected for the first yard-bay collection in order to accomplish the partial tour quantity requirement.

Chapter 4

Numerical Tests

This chapter will present comparative test results for the single straddle carrier routing problem.

Four sets of numerical tests have been performed over the implemented strategies Beam Search and Ant Colony Optimization. The first one, Preliminary Tests, analyzes the behavior of the parameters and their impact on the solution and it also makes a comparison between the Greedy and the Random container collection strategies. The second one, Random Tests, analyzes solutions quality and execution time trends. The third and the forth ones, Benchmark Tests and Practical Tests, compares the obtained results with published results.

All tests have been performed using a Linux workstation (512MB RAM and 1GHz processor).

Table 4.1: PT - Block Distances Matrix

Blocks	block-1	block-2	block-3
block-1	0	100	190
block-2	100	0	100
block-3	190	100	0

Table 4.2: PT - Work Schedule

ContainerGroup	CG1	CG2	CG1	CG2	CG1	CG2
Quantity	10	8	15	10	20	12

4.1 Preliminary Tests (PT)

4.1.1 Input

The input used in the Preliminary Tests (PT) have been retrieved from [11]. They are defined in the following way:

There are three yard-bays and each yard-bay is located in a separate block. So yard-bay 1 is located on block-1, yard-bay 2 is located on block-2 and yard-bay 3 is located on block-3. The distances between the blocks can be found in Table 4.1, the work schedule is given by Table 4.2 and the distribution of containers is given by Table 4.3.

It is assumed that initially the SC is located at block-1/yard-bay 1.

Table 4.3: PT - The Distribution of the Containers

ContainerGroup	CG1	CG2
yard-bay 1	18	15
yard-bay 2	11	7
yard-bay 3	16	8

4.1.2 Parameter values

The Beam Search heuristic has only one parameter to be set which can impact solutions quality. This parameter is the width of the search tree to be explored, called beam width. The Ant Colony heuristic has several parameters to be set and therefore a variety of parameter combinations. These preliminary tests have the purpose of analyzing the Beam Search problem solutions upon a variety of beam widths and the Ant Colony results against different parameter sets.

The results returned by the Beam Search and Ant Colony heuristics will be compared with the optimal solutions provided in [11].

4.1.2.1 Beam Search Results - Greedy container collection strategy

Table 4.4 contains the cost and time response returned by the Beam Search heuristic, against the optimal cost. Table 4.5 contains the visiting routes returned by the Beam Search heuristic, against the optimal route.

By analyzing the results in tables 4.4 and 4.5, a few remarks can be considered:

1. The Beam width up to 50 resulted in the same cost and the same solution

Table 4.4: PT - Beam Search results: costs and processing times

Beam Width	**Optimal**	1	2	10	50	100	500	1000	5000	10000
Cost	**400**	550	550	550	550	500	500	500	500	500
Time (sec)		0.02	0.02	0.05	0.3	1	22	93	1117	2778

Table 4.5: PT - Beam Search results: visiting routes

Cost	ContainerGroups	CG1	CG2	CG1	CG2	CG1	CG2
	WORK SCHEDULE	10	8	15	10	20	12
400	**Visiting Sequence**	**yb1**	**yb1**	**yb1, yb2**	**yb2, yb3**	**yb3, yb2**	**yb2, yb1**
	Quantity*	**(10)8**	**(8)7**	**(8)0, (7)4**	**(2)5, (8)0**	**(16)0, (4)0**	**(5)0, (7)0**
550	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb3, yb1
	Quantity*	(10)8	(8)7	(8)0, (7)4	(7)0, (3)5	(16)0, (4)0	(5)0, (7)0
500	Visiting Sequence	yb3	yb3	yb3, yb1	yb1	yb1, yb2	yb2, yb1
	Quantity*	(10)6	(8)0	(6)0, (9)9	(10)5	(9)0, (11)0	(7)0, (5)0

* A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later.

route as the beam width equal to 1, which corresponds to a greedy search where the best alternative is always selected. In those cases, only a small amount of best solutions has been kept under consideration and the search space has not been well explored. Better solutions were found with the Beam width equal to 100 or greater because more alternatives were explored, however the processing time increased considerably.

2. Beam Search did not find the optimal solution or a solution close to it, even with large beam widths such as 5K or 10K. The implemented heuristic looks for the best route but applies the greedy approach when deciding about how many containers to collect at each yard bay. The optimal solution has chosen (for the fourth partial tour) to collect just 2 containers at yard-bay 2, instead of collecting all 7 containers available, and has chosen to collect all 8 containers available when visiting yard-bay 3. In practice, this route has not even been considered by the Beam Search heuristic method, with the Greedy container collection approach, since this search method always looks for the maximum number of containers which can be picked up at each yard-bay.

Table 4.6: PT - Beam Search with Random collection strategy: cost and processing times

Beam Width	**Optimal**	1	2	10	50	100	500	1000	5000	10000
Cost	**400**	700	550	550	550	500	500	500	400	550
Time (sec)		0,004	0,006	0,03	0,3	1	17	74	989	2451
Solution #		1	2	2	2	3	4	4	5	6

4.1.2.2 Beam Search Results - Random container collection strategy

As discussed at the end of the previous subsection we realized that the Greedy collection strategy is very limited. For the problem under investigation the optimal solution will never be reached, no matter how much the beam width is expanded. Therefore the Random container collection strategy, described in the previous chapter, is tested.

Table 4.6 contains the cost and time responses returned by the Beam Search with the Random collection approach, as well as the optimal cost. Table 4.7 contains the visiting routes returned by Beam Search heuristic against the optimal route.

By analyzing the results given in tables 4.6 and 4.7, a few remarks can be made:

1. The Random collection approach allowed Beam Search to find the optimal solution, what was impossible using the Greedy container collection approach.

Table 4.7: PT - Beam Search with Random collection strategy: visiting routes

Cost	CGroups	Solution #	CG1	CG2	CG1	CG2	CG1	CG2
	WSCHEDULE		10	8	15	10	20	12
400	**Vis.Sequence**		**yb1**	**yb1**	**yb1, yb2**	**yb2, yb3**	**yb3, yb2**	**yb2, yb1**
	Quantity*		**(10)8**	**(8)7**	**(8)0, (7)4**	**(2)5, (8)0**	**(16)0, (4)0**	**(5)0, (7)0**
700	Vis.Sequence	1	yb1	yb1	yb3	yb3, yb2	yb2, yb3, yb1	yb1, yb2, yb3
	Quantity*		(10)8	(8)7	(15)1	(6)2, (4)3	(11)0, (1)0, (8)0	(7)0, (3)0, (2)0
550	Vis.Sequence	2	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb2, yb3, yb1
	Quantity*		(10)8	(8)7	(8)0, (7)4	(4)3, (6)2	(16)0, (4)0	(3)0, (2)0, (7)0
550	Vis.Sequence	6	yb2	yb2, yb1	yb1	yb1	yb1, yb2, yb3	yb3, yb1
	Quantity*		(10)1	(7)0, (1)14	(15)3	(10)4	(3)0, (1)0, (16)0	(8)0, (4)0
500	Vis.Sequence	3	yb1	yb1	yb1, yb2	yb2, yb1	yb1, yb3	yb3, yb1
	Quantity*		(10)8	(8)7	(4)4, (11)0	(7)0, (3)4	(4)0, (16)0	(8)0, (4)0
500	Vis.Sequence	4	yb3	yb3	yb3, yb1	yb1	yb1, yb2	yb2, yb1
	Quantity*		(10)6	(8)0	(6)0, (9)9	(10)5	(9)0, (11)0	(7)0, (5)0
400	Vis.Sequence	5	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb2, yb1
	Quantity*		(10)8	(8)7	(8)0, (7)4	(2)5, (8)0	(16)0, (4)0	(5)0, (7)0

* A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later.

2. The optimal solution, however, has not always been reached because the search tree only deals with route alternatives and not with quantity alternatives. So, at each selected route alternative, a random "possible" quantity is chosen, what does not necessarily lead to a good solution. In terms of route alternatives, the solution should have been reached with beam width greater than or equal to 1000 but the quantity alternatives leaves us on an absolutely random situation, with no guarantees. Although the random container collection strategy has allowed the optimal solution to be reached, quantity alternatives are not explored, due to the structure of the selected Beam Search method.

4.1.2.3 Ant Colony Optimization Results

Four sets of tests were performed in order to find out a good combination of the parameter values. In each set, the value of only one parameter was varied while the other values were kept fixed.

FIRST SET

In this set of tests, the parameter to be varied is the number of ants. The following fixed values were selected for other parameters:

- $nIterations = 15$
- $\alpha = 0.1$, $\beta = 1$, $q_0 = 0.9$, $\tau_0 = 0.005$, $\rho_0 = 0.1$

Table 4.8: PT - Ant Colony Optimization with the Greedy collection strategy: costs and processing times (different numbers of ants)

Ants #	**Optimal**	1	2	10	50	100
Cost	**400**	550	550	500	500	500
Time (sec)		0,05	0,2	4,8	153	619
Solution found at		i=1;a=1	i=1;a=1	i=11;a=1	i=14;a=19	i=8;a=12

Table 4.9: PT - Ant Colony Optimization with the Random collection strategy: costs and processing times (different numbers of ants)

Ants #	**Optimal**	1	2	10	50	100	150
Cost	**400**	500	500	400	450	400	400
Time (sec)		0,06	0,2	5,8	183	722	1658
Solution found at		i=3;a=1	i=11;a=1	i=6;a=4	i=3;a=18	i=15;a=14	i=3;a=123

Table 4.8 contains the cost and time responses returned by the Ant Colony heuristic with the Greedy container collection strategy in addition to the optimal cost. Table 4.9 contains the same results obtained by the Ant Colony Optimization with the Random container collection strategy.

Table 4.10 contains the visiting routes returned by the Ant Colony heuristic with both collection strategies, as well as the optimal route.

By analyzing the results in tables 4.8, 4.9 and 4.10 the following remarks can be made:

1. Ant Colony Optimization using the Random collection strategy demonstrated to return solutions of better quality than with the Greedy collection

Table 4.10: PT - Ant Colony Optimization with Greedy and Random collection strategies: visiting routes

Cost	ContainerGroups	CG1	CG2	CG1	CG2	CG1	CG2
	WORK SCHEDULE	10	8	15	10	20	12
400	**Visiting Sequence**	**yb1**	**yb1**	**yb1, yb2**	**yb2, yb3**	**yb3, yb2**	**yb2, yb1**
	Quantity*	**(10)8**	**(8)7**	**(8)0, (7)4**	**(2)5, (8)0**	**(16)0, (4)0**	**(5)0, (7)0**
550	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb1	yb2, yb3	yb3, yb1
	Quantity*	(10)8	(8)7	(8)0, (7)4	(7)0, (3)4	(4)0, (16)0	(8)0, (4)0
500	Visiting Sequence	yb1	yb1	yb1, yb2	yb1, yb2	yb2, yb3	yb3, yb2
	Quantity*	(10)8	(8)7	(8)0, (7)4	(7)0, (3)4	(4)0, (16)0	(8)0, (4)0
450	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb1	yb1, yb3	yb3, yb2
	Quantity*	(10)8	(8)7	(4)4, (11)0	(3)4, (7)0	(4)0, (16)0	(8)0, (4)4
400	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb2, yb1
	Quantity*	(10)8	(8)7	(8)0, (7)4	(2)5, (8)0	(16)0, (4)0	(5)0, (7)0

* A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later.

approach, within the same range of the processing time.

2. Due to the random nature of this heuristic method, allowing more ants to look for solutions does not necessarily guarantee that a better solution will be reached. As observed in table 4.9, 10 ants found a better quality solution than 50 ants.

3. By looking at tables 4.6 and 4.9, we notice that Ant Colony Optimization worked better than Beam Search with the Random collection strategy since it found the optimal solution more often. In fact, it also found alternative solutions such as a route with cost 450. It can be explained by the fact that the Ant Colony Optimization selects the next location based on clusters,

i.e. both the yard-bay and the quantity of containers are taken into account, while the Beam Search selects the next location based on the yard-bays only. After this selection it applies the random collection strategy (see Remark 2 in the previous subsection).

4. From now on no tests will be performed using the greedy collection strategy because of the poor results obtained with it.

SECOND SET

In this case, the effect of the number of iterations is studied. The following fixed parameter values were used:

- $nAnts = 10$
- $\alpha = 0.1$, $\beta = 1$, $q_0 = 0.9$, $\tau_0 = 0.005$, $\rho_0 = 0.1$

Table 4.11 contains cost and time responses returned by the Ant Colony heuristic with the Random container collection strategy in addition to the optimal cost. Table 4.12 contains the visiting routes returned by the Ant Colony heuristic using the same collection strategy, as well as the optimal route.

By analyzing result tables 4.11 and 4.12, the following remark can be made:

1. It seems that increasing the number of iterations has the same effect as increasing the number of ants looking for solutions but, in fact, there is a

Table 4.11: PT - Ant Colony Optimization with Random collection strategy: costs and processing times (different numbers of iterations)

nIterations	**Optimal**	10	15	20	50	100	150
Cost	**400**	450	450	450	450	400	400
Time (sec)		2	6	10	76	317	743
Solution found at		i=2;a=5	i=9;a=1	i=3;a=7	i=13;a=2	i=16;a=5	i=9;a=1

Table 4.12: PT - Ant Colony Optimization with Random collection strategy: visiting routes (different numbers of iterations)

Cost	ContainerGroups	CG1	CG2	CG1	CG2	CG1	CG2
	WORK SCHEDULE	10	8	15	10	20	12
400	**Visiting Sequence**	**yb1**	**yb1**	**yb1, yb2**	**yb2, yb3**	**yb3, yb2**	**yb2, yb1**
	Quantity*	**(10)8**	**(8)7**	**(8)0, (7)4**	**(2)5, (8)0**	**(16)0, (4)0**	**(5)0, (7)0**
450	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb1	yb1, yb3	yb3, yb2
	Quantity*	(10)8	(8)7	(4)4, (11)0	(3)4, (7)0	(4)0, (16)0	(8)0, (4)4
400	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb2, yb1
	Quantity*	(10)8	(8)7	(8)0, (7)4	(2)5, (8)0	(16)0, (4)0	(5)0, (7)0

* A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later.

difference which did not come out for this small size problem. At each iteration, a global pheromone update occurs and the pheromone is deposited on the edges of the best solution. At each ant solution built, a local pheromone update happens and the pheromone is deposited on the edges of the ant solution. So more ants means more local pheromone updates and therefore the search space is expanded since lots of edges receive pheromone. By fixing the number of ants and increasing the number of iterations, the search space is reduced since only the current best solution is granted with extra pheromone. So, in theory, increasing the number of ants should be more promising than increasing the number of iterations. However there are other parameters, such as the control parameter β, which also contribute.

THIRD SET

Under this third set of preliminary test all parameters will be kept constant, except the parameter β. The following values were used:

- $nAnts = 50$; $nIterations = 50$
- $\alpha = 0.1$, $q_0 = 0.9$, $\tau_0 = 0.005$, $\rho_0 = 0.1$

As explained in section 3, the parameter β weights the importance of selecting short edges in relation to selecting edges with a large amount of pheromone. Thus it measures how much diversity will be allowed in the search space and is used for determining a candidate yard-bay probability, defined as in Formula 4.1.

$$probability = pheromone * (1/distance(currentLocation, candidateLocation))^{\beta} \quad (4.1)$$

In the first and second test sets β value was set equal to 1. Now we will study the following cases and track the quality if the solutions:

- $\beta \in \{0, 1, 2\}$
- $\Delta\ \beta \in \{\frac{1}{nIterations}, \frac{2}{nIterations}\}$. At each run, $\beta \leftarrow \beta + \Delta\beta$
- $\beta = 0$ for the first run and $\beta \in \{1, 2\}$ for all other runs

Table 4.13 contains cost and time responses returned by the Ant Colony heuristic with the Random container collection strategy upon the variation of the value of β. Table 4.14 contains the corresponding visiting routes.

An analysis of the results in tables 4.13 and 4.14, leads us to make the following remarks:

1. According to Formula 4.1, if β is zero, only the pheromone is considered and the yard-bay distances are ignored. That is the reason why a worse solution has been reached with the value $\beta = 0$ for the problem under analysis, even though no real difference has been observed in the processing time. If β is greater than zero, the pheromone and the inverse

Table 4.13: PT - Ant Colony Optimization with Random collection strategy: costs and processing times (different β)

Beta	**Optimal**	$\beta = 0$	β=1	$\beta = 2$	$\Delta\ \beta = 1$	$\Delta\ \beta = 2$
Cost	**400**	500	400	400	400	400
Time (sec)		1984	2000	2028	2063	2008
Solution found at		i=5;a=29	i=10;a=25	i=1;a=39	i=11;a=2	i=15;a=26

Beta	**Optimal**	$\Delta\ \beta = 3$	$\beta = 0 \vert \beta = 1$	$\beta = 0 \vert \beta = 2$	$\beta = 0 \vert \beta = 3$
Cost	**400**	400	400	400	400
Time (sec)		2036	2153	2004	2000
Solution found at		i=3;a=48	i=7;a=26	i=5;a=18	i=39;a=29

Table 4.14: PT - Ant Colony Optimization with Random collection strategy: visiting routes (different β)

Cost	ContainerGroups	CG1	CG2	CG1	CG2	CG1	CG2
	WSCHEDULE	10	8	15	10	20	12
400	**Vis.Sequence**	**yb1**	**yb1**	**yb1, yb2**	**yb2, yb3**	**yb3, yb2**	**yb2, yb1**
	Quantity*	**(10)8**	**(8)7**	**(8)0, (7)4**	**(2)5, (8)0**	**(16)0, (4)0**	**(5)0, (7)0**
500	Vis.Sequence	yb1	yb1	yb1, yb2	yb1, yb2	yb2, yb3	yb3, yb2
	Quantity*	(10)8	(8)7	(8)0, (7)4	(7)0, (3)4	(4)0, (16)0	(8)0, (4)0
400	Visiting Sequence	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb2, yb1
	Quantity*	(10)8	(8)7	(8)0, (7)4	(2)5, (8)0	(16)0, (4)0	(5)0, (7)0

* A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later.

of the distance will be taken into account. So, considering that the value $1/distance(currentLocation, candidateLocation)$ is greater than zero and at most equal to one, we observe that in increasing β the probability reduces. The same happens for large distances which also have a reducing effect on probability. Having in mind that the Ant Colony algorithm rewards larger probabilities and, considering β unchanged, distance will be the factor which will really make the difference. All remaining tests will be performed with β set to 2.

2. No visible difference was observed for this small size problem, when using the three different β strategies, i.e. single β strategy, $\Delta\beta$ strategy and β zero in the start and one of the values 1 or 2 later strategy.

FOURTH SET

Under this third set of preliminary test all parameters will be kept constant, except the value of the exploration/exploitation control parameter q_0. The fixed values are now:

- $nAnts = 50$; $nIterations = 50$
- $\alpha = 0.1$, $\beta = 2$, $\tau_0 = 0.005$,$\rho_0 = 0.1$

As explained in section 3, the parameter q_0 defines the chance a search method (exploration or exploitation) will have, when choosing the next yard-bay to visit.

Table 4.15: PT - Ant Colony Optimization with Random collection strategy: costs and processing times (different q_0)

q0	**Optimal**	$q_0 = 0.1$	$q_0 = 0.5$	$q_0 = 0.9$
Cost	**400**	400	400	400
Time (sec)		2125	2078	2146
Solution found at		i=1;a=34	i=1;a=16	i=25;a=17

Table 4.16: PT - Ant Colony Optimization with Random collection strategy: visiting routes (different q_0)

Cost	ContainerGroups	CG1	CG2	CG1	CG2	CG1	CG2
	WSCHEDULE	10	8	15	10	20	12
400	**Vis.Sequence**	**yb1**	**yb1**	**yb1, yb2**	**yb2, yb3**	**yb3, yb2**	**yb2, yb1**
	Quantity*	**(10)8**	**(8)7**	**(8)0, (7)4**	**(2)5, (8)0**	**(16)0, (4)0**	**(5)0, (7)0**
400	Vis.Sequence	yb1	yb1	yb1, yb2	yb2, yb3	yb3, yb2	yb2, yb1
	Quantity*	(10)8	(8)7	(8)0, (7)4	(2)5, (8)0	(16)0, (4)0	(5)0, (7)0

* A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later.

If the random number $q \in [0, 1]$ is less than or equal to q_0, the next yard-bay is selected through exploration, i.e. the largest probability value calculated with Formula 4.1 is chosen, otherwise the next yard-bay is selected through exploitation, i.e. the random probability value calculated with Formula 4.1 is chosen.

Table 4.15 contains cost and time responses returned by the Ant Colony heuristic with the Random container collection strategy upon q_0 variation. Table 4.16 contains the corresponding visiting routes.

By analyzing result tables 4.15 and 4.16, the following remark can be made:

1. In our tests preferring the random selection $(q_0 = 0.1)$ in favor of exploitation or giving equal chances to deterministic and random selection $(q_0 = 0.5)$, we observed that the optimal solution was reached much earlier than in the case of preferring the deterministic selection in favor of exploration $(q_0 = 0.9)$. In the first two cases the optimal solution was found during the first iteration, while in the third case 25 iterations were needed for this. All remaining tests will be performed with q_0 set to 0.5.

4.2 Random Tests (RT)

4.2.1 Overview

The second series of our tests is organized as presented in [9]. In it, the number of blocks used (NBU) is varied and, for each number, a random distribution at containers is created.

The approach consists of randomly generating the distribution of containers under NBU (number of blocks used) blocks. That is, the containers from each container group are dispersed over the port yard according to another parameter varied in the tests, MNBAG, the maximum number of blocks allocated for each container group. Eighteen combinations of NBU and MNBAG values are used, as described below.

- NBU = 2; MNBAG = 1: container groups are randomly distributed on two blocks but each group is concentrated on a single block. This means that only the distribution of containers within each group inside a block is randomly generated.

  An example:

  10 containers belong to container group A and 7 belong to group B

  1. 10 containers from container group A are located at a randomly selected unique block, say block-1
  2. 7 containers from container group B are located at a randomly selected unique block, say block-2

- NBU = 2; MNBAG = 2: container groups are randomly distributed on two blocks and each group is necessarily dispersed over two blocks.

  An example:

  10 containers belong to container group A and 7 belong to group B

  1. A randomly selected number of containers, say 6 containers from group A, are located on block-1
  2. A randomly selected number of containers, say 4 containers from group A, are located on block-2

3. A randomly selected number of containers, say 2 containers from group B, are located on block-1
4. A randomly selected number of containers, say 5 containers from group B, are located on block-2

- NBU = 2; MNBAG = *random*: container groups are randomly distributed on two blocks and each group is randomly dispersed over one or two blocks.

  An example:

  10 containers belong to container group A and 7 belong to group B

  1. A randomly selected number of containers, say 10 containers from group A, are located at a randomly selected number of blocks, say block-1
  2. A randomly selected number of containers, say 3 containers from group B, are located at a randomly selected number of blocks, say block-1
  3. A randomly selected number of containers, say 4 containers from group B, are located at a randomly selected number of blocks, say block-2

- NBU = 3; MNBAG = 1, 2, 3, *random*: basically the same process as described for NBU = 2;

Table 4.17: RT - Container Groups

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
Quantity	22	13	31	11	27	9	37	15

Table 4.18: RT - Work Schedule

CG	CG3	CG5	CG7	CG2	CG6	CG1	CG8	CG7	CG8	CG2	CG4	CG1	CG3
Quant	12	27	23	4	9	8	12	14	3	9	11	14	19

- NBU = 4; MNBAG = 1, 2, 3, 4, *random*: basically the same process as described for NBU = 2;
- NBU = 5; MNBAG = 1, 2, 3, 4, 5, *random*: basically the same process as described for NBU = 2.

4.2.2 Input

The overall task of the straddle carrier consists of moving 165 containers that belong to eight different container groups (Table 4.17).

This unique SC is assigned to a unique quay crane. The SC Work Schedule, to be followed exactly, is given in Table 4.18.

Each block contains 22 yard-bays and the distance between two side-by-side yard-bays is equal to 3 meters. The distances between the blocks in meters are defined in Table 4.19.

Table 4.19: RT - Distances between the blocks

Blocks	block-1	block-2	block-3	block-4	block-5
block-1	0	100	190	280	370
block-2	100	0	100	190	280
block-3	190	100	0	100	190
block-4	280	190	100	0	100
block-5	370	280	190	100	0

The SC starting point is assumed to be block-1/yard-bay 11.

4.2.3 Beam Search and Ant Colony Results

The parameter values for the Ant Colony heuristic used in these tests were selected on the basis of the preliminary test described in Section 4.2. Thus it was selected ant parameters: $nIterations = 15; \alpha = 0.1; \beta = 2; q0 = 0.5; \tau_0 = 0.005; \rho_0 = 0.1$.

Each of the eighteen combinations of NBU and MNBAG were run 20 times.

For both heuristics, the MNBAG x cost values were plotted for different values of NBU (Figures 4.1 and 4.2) using beam width and the number of ants equal to 10, respectively . The last point of each curve, for which MNBAG=NBU+1, corresponds to the case MNBAG=*random*.

On the basis of the graphics in Figures 4.1 and 4.2 we can consider that:

- The cost increases with increasing MNBAG for all NBU values, i.e. the more scattered containers are within a yard the higher is the route cost. In

Figure 4.1: RT - Beam Search MNBAG x cost

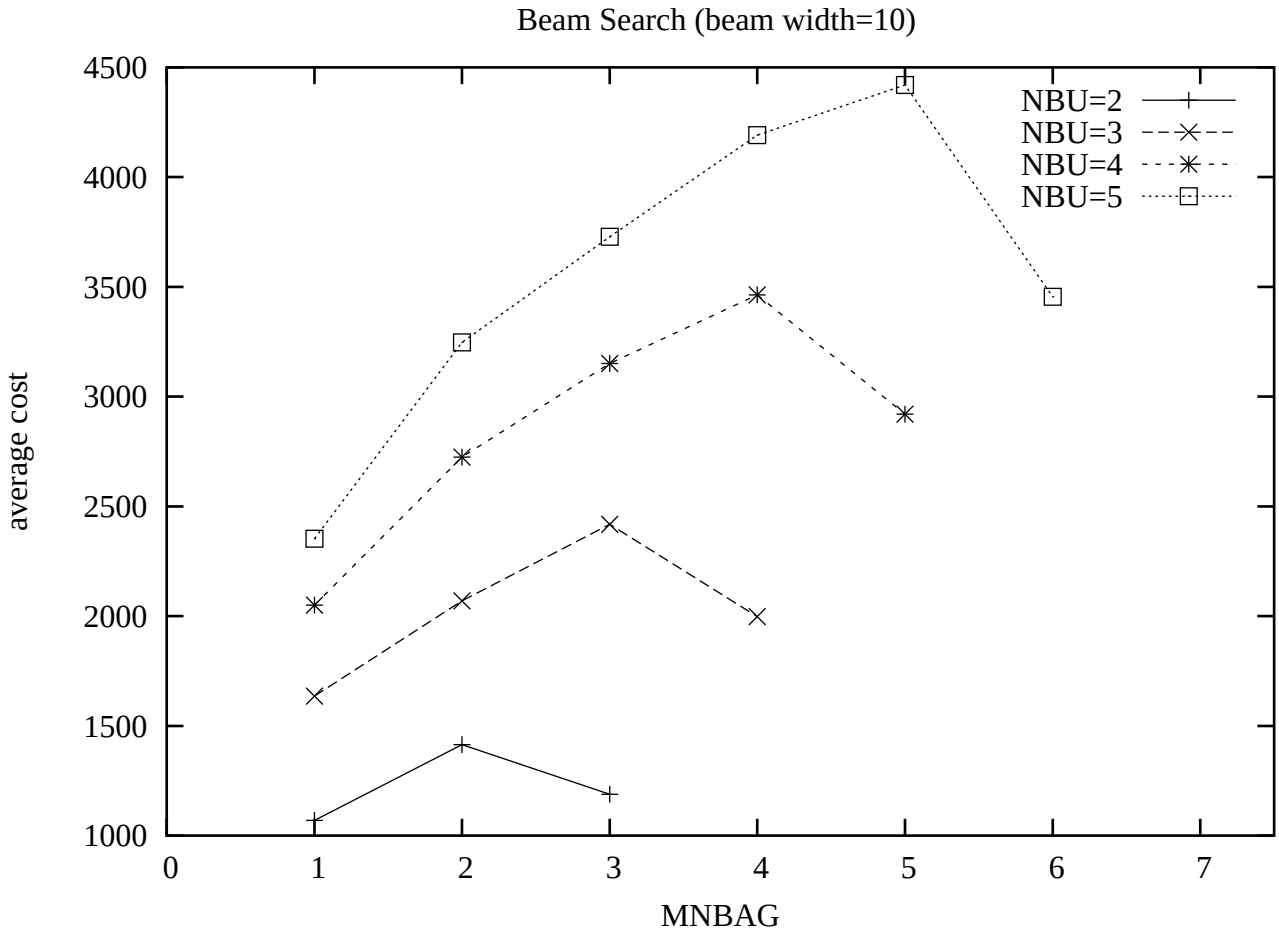
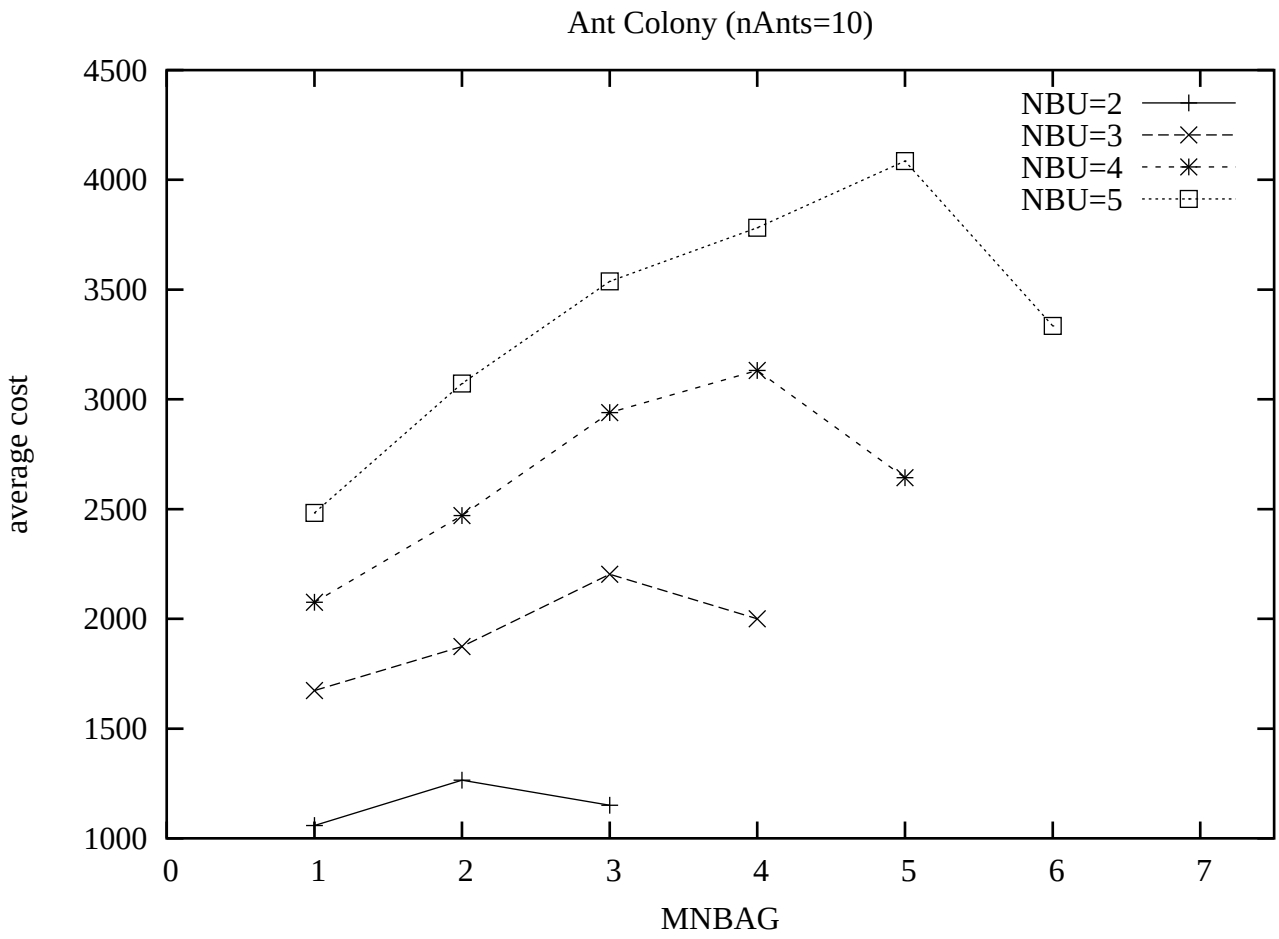


Figure 4.2: RT - Ant Colony heuristic MNBAG x cost



average, MNBAG=*random* cases showed costs lower than the case MNBAG=NBU.

- Ant Colony heuristic tends to find, in average, solutions of lower cost than Beam Search heuristic.

Figures 4.3 and 4.4 represent the average processing times and costs obtained by the two heuristics with values NBU=MNBAG=5.

Analyzing graphics in Figures 4.3 and 4.4 we can consider that:

- With respect to the processing times, Ant Colony heuristic is worse than Beam Search heuristic. We can observe that the time is approximately an exponential function of the number of ants.

- Ant Colony heuristic returns better solutions. If ignoring the cases in which the number of ants and the beam widths were below or equal to 5, we can say that the Beam Search found solutions with costs oscillating between 4250 and 4450 while the Ant Colony heuristic found costs oscillating between 3900 and 4100.

- Larger beam widths, i.e. 25 and 30, returned poorer quality solutions than beam widths 15 and 20. An opposite behavior was observed with Ant Colony heuristic, since 10 or 15 ants found worse or equal solution costs comparing to 25 and 30. So as the number of ants increased, Ant Colony

Figure 4.3: RT - Beam Search and Ant Colony heuristic: processing times

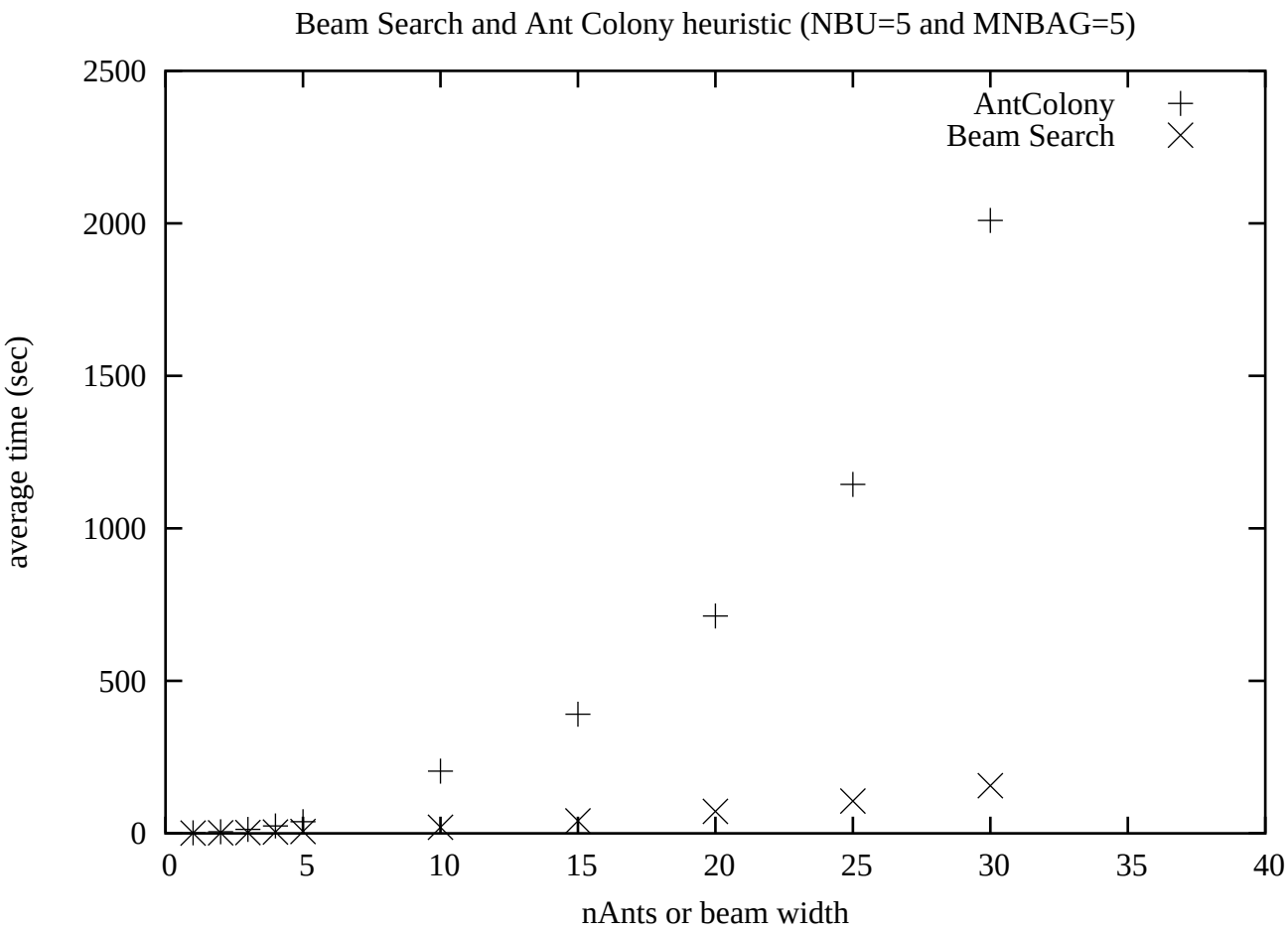
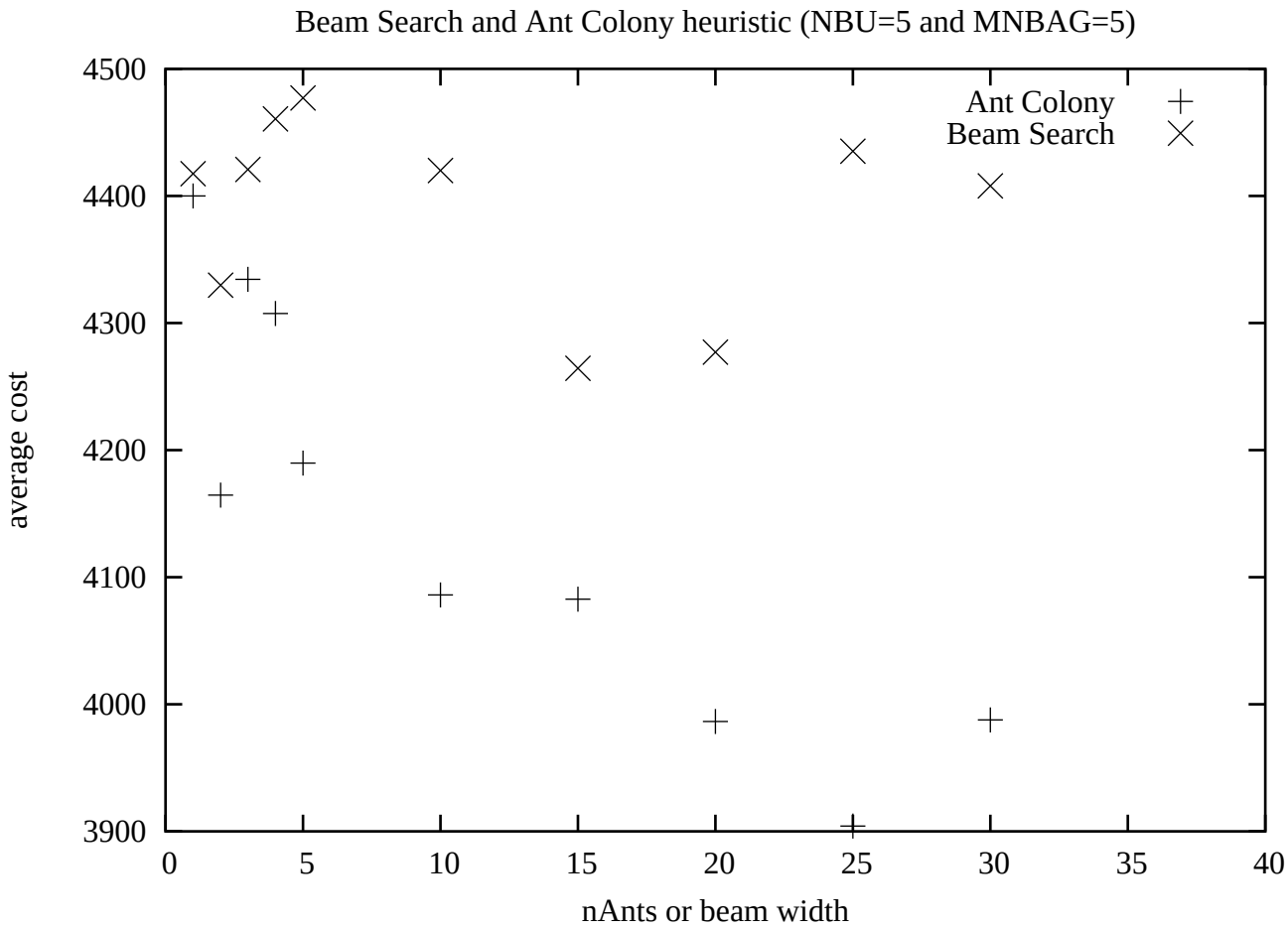


Figure 4.4: RT - Beam Search and Ant Colony: costs of the best solutions



heuristic tended to produce better results however its average processing time became really high (around 2000min = 33h), compared to Beam Search (around 200min = 3.3h).

4.3 Benchmark Tests (BT)

4.3.1 Overview

The benchmark tests described in [11] were recreated. They consist of eight small-size problems each having a container distribution pattern as shown in Tables 4.20, 4.21, 4.22, 4.23, 4.24, 4.25, 4.26 and 4.27.

These problems assume that the containers are located on the central yard-bay of each block. Therefore, since the blocks have 22 yard-bays each, the containers are located on yard-bay 11 for block-1, on yard-bay 33 for block-2 and on yard-bay 55 for block-3.

Results returned by Beam Search and Ant Colony heuristics will be compared with solutions provided in [11]. These were computed using Mixed Integer Programming.

Table 4.20: BT - Problem #1 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	16		10	7				6
block-2		12	10	4	10	5	24	3
block-3	6	1	11		17	4	13	6

Table 4.21: BT - Problem #2 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1		7	13				17	5
block-2	12			4	26	4		10
block-3	10	6	18	7	1	5	20	

Table 4.22: BT - Problem #3 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	13		6			7	22	5
block-2	9	5	7	4	26	2	15	10
block-3		8	18	7	1			

Table 4.23: BT - Problem #4 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	7		14	3	9	3	14	
block-2	10	6	11		8	6	17	8
block-3	5	7	6	8	10		6	7

Table 4.24: BT - Problem #5 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	15	2	13		16		9	
block-2	3	5	11	4		4	13	4
block-3	4	6	7	7	11	5	15	11

Table 4.25: BT - Problem #6 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	12	2	10	4	10		10	5
block-2	6	5	11	7	11	4	13	4
block-3	4	6	10		6	5	14	6

Table 4.26: BT - Problem #7 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	10	2	8	4	10	2	13	4
block-2	6	9	14	4	11	2	9	5
block-3	6	2	9	3	6	5	15	6

Table 4.27: BT - Problem #8 Distribution of Containers

Container Group	CG1	CG2	CG3	CG4	CG5	CG6	CG7	CG8
block-1	12	2	2	4	6	3	10	1
block-2	3	5	19	2	11	4	13	8
block-3	7	6	10	5	10	2	14	6

4.3.2 Input

The quantities of the containers in each group are the same used for the Random Tests (Table 4.17).

Again there is one SC assigned to one QC and the Work Schedule already presented for Random Tests (Table 4.18) is assumed.

Table 4.1 indicates the distances between the blocks. There are three blocks. The side-by-side yard-bays are 3 meters apart.

The SC starting point is assumed to be block-1/yard-bay 1.

4.3.3 Beam Search and Ant Colony Results

Tables 4.28 and 4.29 show the route costs taken from [11] and the results obtained with the Beam Search and Ant Colony heuristics using the Random collection strategy. Average, best and worse colums refer to results from 5 runs for each parameter set.

Upon the number of ants variation, the constant parameter values will be set as follows:

- $nIterations = 15; \alpha = 0.1; \beta = 2; q_0 = 0.5; \tau_0 = 0.005; \rho_0 = 0.1$

The following observations can be made:

1. The Ant Colony heuristic returned lower route costs than Beam Search.

Table 4.28: BT Beam Search processing time x cost

problems #	Mixed Integer	beam search								
	Programming	30			100			500		
	from [11]	average	best	worse	average	best	worse	average	best	worse
problem 1	**1931**	2292	2166	2419	2238	**1930**	2419	2175	2033	2346
problem 2	**1961**	2263	2093	2466	2225	2020	2466	2109	**1960**	2526
problem 3	**1841**	2058	1853	2179	2179	2179	2179	2046	1853	2179
problem 4	**2347**	2492	2492	2492	2542	2492	2745	2552	2419	2685
problem 5	**2420**	2918	2745	3071	2751	2672	2925	2889	2745	2925
problem 6	**2330**	2928	2745	3144	2800	2655	3071	2771	2672	2938
problem 7	**2656**	3141	3071	3337	3037	3011	3084	3052	3011	3084
problem 8	**not solved**	2948	2818	3071	2918	2758	3161	2951	2758	3264

Table 4.29: BT Ant Colony heuristic processing time x cost

problems #	Mixed Integer	ant colony								
	Programming	30			50			100		
	from [11]	average	best	worse	average	best	worse	average	best	worse
problem 1	**1931**	2100	2003	2166	2075	**1930**	2166	2009	**1930**	2093
problem 2	**1961**	2020	2020	2020	1996	**1960**	2020	1996	**1960**	2020
problem 3	**1841**	1922	1853	2020	1854	**1840**	1913	1854	**1840**	1913
problem 4	**2347**	2462	2419	2492	2448	2419	2492	2433	2419	2492
problem 5	**2420**	2535	2492	2565	2542	2492	2672	2477	**2419**	2492
problem 6	**2330**	2733	2685	2745	2629	2419	2745	2596	2492	2745
problem 7	**2656**	3027	2908	3161	3053	2981	3071	2951	2908	2981
problem 8	**not solved**	2792	2582	2908	2738	2655	2818	2687	2655	2818

2. By increasing the number of ants we obtain better lower cost solutions because the search space is better explored. This behavior confirmed the tendency observed in previous tests.

3. Reaching the optimal solution is not guaranteed even for medium size problems. The difference of one between the results obtained by Mixed Integer Programming from [11] and the results obtained with the Beam Search and Ant Colony heuristic for problems #1, #2, #3 and #5 was considered irrelevant.

4. Heuristic methods Beam Search and Ant Colony reached a solution for problem #8, not obtained by Mixed Integer Programming.

4.4 Practical Tests (RT)

4.4.1 Overview

Two practical problems defined in [11] have been reproduced. Next two subsections present the inputs and results of each problem.

Table 4.30: RT1 - Work Schedule

C.Group	HAM40	HAM20	RTM20	HAM20	SOU20	JED20	SOU20
Quantity	6	9	10	4	28	7	6

Table 4.31: RT1 - Containers Distribution Table

blocks	yard-bay	ContainerGroup	Quantity
block-1	2	SOU20	10
block-1	3	SOU20	9
block-1	4	SOU20	5
block-1	4	JED20	4
block-1	7	HAM40	4
block-1	8	HAM40	2
block-2	2	SOU20	10
block-2	6	JED20	3
block-3	1	RTM20	5
block-3	2	RTM20	5
block-3	4	HAM20	9
block-3	6	HAM20	4

4.4.2 Practical problem #1 (RT1)

4.4.2.1 Input

A straddle carrier (SC1) is assigned to a Quay Crane (QC1) in this problem. It has the overall task to load 70 containers classified under five container groups. The work schedule is given by Table 4.30 and the distribution of the containers is given by Table 4.31.

The distance between adjacent yard-bays is 3m and between adjacent blocks is 100m. The container yard is divided into three blocks, each one containing 8 yard-bays. The starting point of SC1 is block-1/ yard-bay 1.

4.4.2.2 Results

The SC1 route identified by [11] is the one provided by the Greedy collection strategy and has a cost 614.

The Ant Colony heuristic, using the Random collection strategy, found other routes with smaller costs (514, 511 and 505). The Beam Search, also using the Random collection strategy, returned routes with costs 520 and 517, thus greater than the Ant Colony routes but better than the Greedy collection strategy results. Table 4.33 shows the results.

In the Beam Search, the beam width was equal to 30.

The Ant Colony Optimization parameters were: $nAnts = 30; nIterations = 15; \alpha = 0.1; \beta = 2; q_0 = 0.5; \tau_0 = 0.005; \rho_0 = 0.1$.

4.4.3 Practical problem #2 (RT2)

4.4.3.1 Input

One straddle carrier (SC1) is assigned to one quay crane (QC1) in this problem. The overall task was to load 146 containers classified under five container groups. The work schedule is given by Table 4.34 and the distribution of the containers is given by Table 4.35.

The distance between adjacent yard-bays is 3m and between adjacent blocks is 100m. The container yard is divided into three blocks: block-1 contains 7 yard-

Table 4.33: RT1 - Beam Search and Ant Colony heuristics: visiting routes

Cost	CG		HAM40	HAM20	RTM20	HAM20	SOU20	JED20	SOU20
	WS		6	9	10	4	28	7	6
614	**VSeq**		**b1/y7, b1/y8**	**b3/y6, b3/y4**	**b3/y2, b3/y1**	**b3/y4**	**b2/y2, b1/y2, b1/y3**	**b1/y4, b2/y6**	**b1/y4, b1/y3**
	Qty*		**(4) 0, (2) 0**	**(4) 0, (5) 4**	**(5) 0, (5) 0**	**(4) 0**	**(10) 0, (10) 0, (8) 1**	**(4) 0, (3) 0**	**(5) 0, (1) 0**
520	VSeq	beam	b1/y7, b1/y8	b3/y6, b3/y4	b3/y2, b3/y1	b3/y4, y3/y6	b1/y2, b1/y3, b2/y2	b2/y6, b1/y4	b1/y4, b1/y3
	Qty*	rand	(4) 0, (2) 0	(3)1, (6)3	(5) 0, (5) 0	(3)0, (1)0	(10)0, (8)1, (10)0	(3)0, (4)0	(5) 0, (1) 0
517	VSeq	beam	b1/y7, b1/y8	b3/y6, b3/y4	b3/y1, b3/y2	b3/y4, y3/y6	b2/y2, b1/y2, b1/y3, b1/y4	b1/y4, b2/y6	b2/y2
	Qty*	rand	(4) 0, (2) 0	(1)3, (8)1	(5) 0, (5) 0	(1)0, (3)0	(4) 6, (10) 0, (9) 0, (5) 0	(4) 0, (3) 0	(6) 0
514	VSeq	ant	b1/y7, b1/y8	b3/y6, b3/y4	b3/y2, b3/y1	b3/y4	b1/y4, b1/y3, b1/y2, b2/y2	b2/y6, b1/y4	b1/y4, b1/y3
	Qty*	rand	(4) 0, (2) 0	(4) 0, (5) 4	(5) 0, (5) 0	(4) 0	(2) 3, (6) 3, (10) 0, (10) 0	(3) 0, (4) 0	(3) 0, (3) 0
511	VSeq	ant	b1/y7, b1/y8	b3/y6, b3/y4	b3/y2, b3/y1	b3/y4	b2/y2, b1/y2, b1/y4, b1/y3	b1/y4, b2/y6	b2/y2
	Qty*	rand	(4) 0, (2) 0	(4) 0, (5) 4	(5) 0, (5) 0	(4) 0	(4) 6, (10) 0, (5) 0, (9) 0	(4) 0, (3) 0	(6) 0
505	VSeq	ant	b1/y7, b1/y8	b3/y6, b3/y4	b3/y2, b3/y1	b3/y4	b2/y2, b1/y2, b1/y3, b1/y4	b1/y4, b2/y6	b2/y2
	Qty*	rand	(4) 0, (2) 0	(4) 0, (5) 4	(5) 0, (5) 0	(4) 0	(4) 6, (10) 0, (9) 0, (5) 0	(4) 0, (3) 0	(6) 0

*A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later

Table 4.34: RT2 - Work Schedule

C.Group	A	B	C	A	D	B	A	E	C
Quantity	15	15	13	7	21	19	16	17	23

bays; block-2 contains 6 yard-bays and block-3 contains 5 yard-bays. SC1 starting point is block-1/ yard-bay 1.

4.4.3.2 Results

In [11], the SC1 route with cost equal to 1314 was identified. The Ant Colony heuristic using the Random container collection strategy of this thesis, found other routes with smaller costs: 853 and 844, as shown in Table 4.36. However, in this particular case, the Greedy collection strategy returned a slightly better route (832). Beam Search returned a better result using the Random collection strategy (cost=841), when compared with the result obtained by the Greedy collection strategy (cost=1044). Table 4.36 shows the results.

In the Beam Search, the beam width was equal to 30.

The Ant Colony Optimization parameters were: $nAnts = 30; nIterations = 15; \alpha = 0.1; \beta = 2; q_0 = 0.5; \tau_0 = 0.005; \rho_0 = 0.1$.

Table 4.35: RT2 - Distribution of the Containers

blocks	yard-bay	ContainerGroup	Quantity
block-1	1	C	9
block-1	2	A	8
block-1	3	A	10
block-1	4	E	4
block-1	5	E	7
block-1	6	B	4
block-1	7	C	10
block-2	8	B	7
block-2	9	A	8
block-2	10	C	7
block-2	10	D	3
block-2	11	B	8
block-2	12	C	10
block-2	13	E	6
block-3	14	A	12
block-3	15	B	6
block-3	16	D	10
block-3	17	B	9
block-3	18	D	8

Table 4.36: RT2 - Beam Search and Ant Colony heuristics: visiting routes

Cost	CG		A	B	C	A	D	B	A	E	C
	WS		15	15	13	7	21	19	16	17	23
1314	**VS**		**y2, y3**	**y6, y8, y11**	**y1, y7**	**y3, y9**	**y10, y16, y18**	**y11, y15, y17**	**y9, y14**	**y4, y5, y13**	**y7, y10, y12**
	Q*		**8(0), 7(3)**	**4(0), 7(0), 4(4)**	**9(0), 4(6)**	**3(4), 4(4)**	**3(0), 10(0), 8(0)**	**4(0), 6(0), 9(0)**	**4(0), 12(0)**	**4(0), 7(0), 6(0)**	**6(0), 7(0), 10(0)**
1044	VS	bg	y2, y14	y15, y17	y12, y10	y9	y10, y18, y16	y8, y11, y6	y3, y14, y9	y13, y5, y4	y7, y1, y10
	Q*		(8)0, (7)5	(6)0, (9)0	(10)0, (3)4	(7)1	(3)0, (8)0, (10)0	(7)0, (8)0, (4)0	(10)0, (5)0, (1)0	(6)0, (7)0, (4)0	(10)0, (9)0, (4)0
853	VS	ar	y2, y3	y6, y8, y11	y10, y12	y9	y10, y18, y16	y15, y17, y8, y11	y9, y14, y3	y4, y5, y13	y12, y7, y1
	Q*		(8)0, (7)3	(4)0, (5)2, (6)2	(7)0, (6)4	(7)1	(3)0, (8)0, (10)0	(6)0, (9)0, (2)0, (2)0	(1)0, (12)0, (3)0	(4)0, (7)0, (6)0	(4)0, (10)0, (9)0
862	VS	ar	y2, y3	y6, y8, y11	y10, y12	y9	y10, y18, y16	y17, y15, y8, y11	y9, y14, y2	y4, y5, y13	y10, y1, y7
	Q*		(5)3, (10)0	(4)0, (5)2, (6)2	(3)4, (10)0)	(7)1	(3)0, (8)0, (10)0	(9)0, (6)0, (2)0, (2)0	(1)0, (12)0, (3)0	(4)0, (7)0, (6)0	(4)0, (9)0, (10)0
844	VS	ar	y2, y3	y6, y8, y11	y10, y12	y9	y10, y18, y16	y15, y17, y11	y9, y14, y3	y4, y5, y13	y12, y7, y1
	Q*		(8)0, (7)3	(4)0, (7)0, (4)4	(7)0, (6)4	(7)1	(3)0, (8)0, (10)0	(6)0, (9)0, (4)0	(1)0, (12)0, (3)0	(4)0, (7)0, (6)0	(4)0, (10)0, (9)0
841	VS	br	y2, y3	y8, y11	y12, y10	y9	y10, y18, y16	y15, y17, y6	y2, y14, y9	y13, y5, y4	y1, y7, y12, y10
	Q*		(5)3, (10)0	(7)0, (8)0	(9)1, (4)3	(7)1	(3)0, (8)0, (10)0	(6)0, (9)0, (4)0	(3)0, (12)0, (1)0	(6)0, (7)0, (4)0	(9(0), (10)0, (1)0, (3)0
832	VS	ag	y2, y3	y6, y11, y8	y10, y12	y9	y10, y18, y16	y17, y15, y8	y9, y14, y3	y4, y5, y13	y12, y7, y1
	Q*		(8)0, (7)3	(4)0, (8)0, (3)4	(7)0, (6)4	(7)1	(3)0, (8)0, (10)0	(9)0, (6)0, (4)3	(1)0, (12)0, (3)0	(4)0, (7)0, (6)0	(4)0, (10)0, (9)0
*A number within parenthesis represents the collected quantity and a number without parenthesis represents the remaining quantity to be collected later											
** bg = beam/greedy; ; br = beam/random; ag = ant/greedy; ar = ant/random											

Chapter 5

Multiple Straddle Carrier Routing Problem

This thesis focused on a single straddle carrier routing problem. However a real container terminal has more than one quay crane, with their respective work schedules, and more than one straddle carrier, each one assigned to one quay crane. Several SCs must complete their routing, by sharing the same container yard-map. This new scenario introduces a few complications to the single SC routing problem, increasing significantly the routing problem complexity.

The present chapter aims to present a discussion about the multiple straddle carrier routing problem in a container terminal environment.

The Multiple SC routing problem has the following constraints to be satisfied:

1. The given work schedules of all QCs must be accomplished;
2. Each SC is assigned to a single QC;
3. The total number of containers from a container group picked up at each yard-bay must be equal to the total number of containers from the same container group initially located at the container terminal yard;
4. Conflicts between SCs must be resolved;
5. Several SC equipments can be working simultaneously;
6. The objective is to minimize each SC total completion time or total travel distance.

A conflict between SCs can be of different types, such as:

- Travel conflict: a SC tries to cross another SC. For example, SC1 is at yard-bay 3 and must pick up containers at yard-bay 1 next. SC2 is picking up containers at yard-bay 2.
- Space conflict: a SC tries to move to the same location where another SC is already placed. For example, SC1 is at yard-bay 1 and must pick up containers at yard-bay 3 next. SC2 is collecting containers at yard-bay 3.

Apart from those two types of conflicts, there is also another important aspect to be considered under the multiple SC routing problem. The container stock at the

container terminal yard must be shared between all SC equipments. It means that if there was initially 10 containers of group A at yard-bay 1 and SC1 picked up 3, SC2 must know in real time that there are only 7 containers of group A left at yard-bay 1.

Kim [11] proposed a job scheduling solution for routing problem of two SCs where containers were located on a single block. The following conflict resolution strategies were considered.

- Travel conflict
  - waiting strategy: SC1, which wants to cross SC2's way, waits until SC2 completes its task and then performs the crossing.
  - exchanging roles between SCs' strategy: SC2's current task is interrupted and 1) SC1 resumes SC2's previously interrupted task and 2) SC2 assumes SC1's task. It means that roles between SCs are exchanged after a job interruption.
- Space conflict
  - waiting strategy: SC1, which wants to move to SC2 location, waits until SC2 finishes its task and then performs the move.
  - substitutive strategy: SC2's current task is interrupted and SC2 assumes SC1's task before resuming its preciously interrupted task.

The routing problem of more than two SCs was also considered in [11] but, this time, the containers were located in one or multiple blocks, according to the assumptions below.

- a pseudo work schedule would be constructed by appending the work schedules of all SCs ;
- no interference between equipments would be considered.

The multiple routing problem has therefore been reduced to a single routing problem. By solving the single SC problem for the pseudo work schedule would, in theory, resolve the overall problem. However assumptions made turn the problem scenario completely artificial, since each SC route will have to be selected manually from the output and each SC routing will have to occur in sequence and never in parallel. So why not using a single SC?

I solved some multiple routing problem, using the single SC routing procedure, by providing (through manual work) the container distribution table for each SC separately. However, this procedure seems inappropriate since the potential parallelism of multiple SCs is ignored.

Chapter 6

Conclusion

The Ant Colony heuristic proved to be more promising than the Beam Search heuristic in spite of the excessive processing times of the former. For example a problem with five container groups, each one dispersed into five yard-bays, took in average 33 hours to be resolved with Ant Colony heuristic against 3.3 hours with Beam Search. Considering that in practice the number of yard-bays where container groups are scattered do not exceed five, according to [11], the heuristic could in fact be used for planning single straddle carrier routes in real container terminals. In case larger problems do need to be solved, more investigation would be required to improve the efficiency of the Ant Colony heuristic.

The proposed Random collection strategy increased the number of alternatives considered within the search space and turned out to improve the quality of solutions. The Beam Search did not react so well as Ant Colony heuristic with

this strategy because it builds its search tree over yard-bay choices making its decisions on the basis of the route costs. Therefore, the list of open states only contains different route choices that is, the beam search does not take account of container quantities. Consequently, it offers no alternatives for a quantity pick-up which turns out to be a bad choice. In contrast, two ants may make the same yard-bay choice with different quantity pick-ups. This means that a larger number of different solutions will be considered during the search.

Solving a single straddle carrier problem does not correspond to real scenarios encountered in container terminals. Thus multiple SC routing problem would be the next natural step for a future work in order to attend to the demands of a port environment. More research in this direction is needed but one promising approach might be the application of parallel programming. The search for a solution should consider the container stock without any separate preallocation of containers to each SC.

Bibliography

[1] Mordecai Avriel, Michal Penna, Naomi Shpirer, and Smadar Witteboon. Stowage planning for container ships to reduce the number of shifts. *Annals of Operations Reasearch*, 1998.

[2] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial optimization*. John Wiley and Sons, Inc., 1998.

[3] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the traveling salesman problem. *Bio Systems*, 1997.

[4] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1997.

[5] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 2003.

[6] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Company, Inc., 2001.

[7] Neil D. Jones. *Computability and complexity: from a programming perspective*. The MIT Press, 1997.

[8] Kap Hwan Kim and Hong Bae Kim. Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics*, 1999.

[9] Kap Hwan Kim and Ki Young Kim. Routing straddle carriers for the loading operation of containers using a beam search algorithm. *Computers and Industrial Engineering*, 1999.

[10] Kap Hwan Kim, Young Man Park, and Kwang-Ryul Ryu. Deriving decision rules to locate expiort containers in container yards. *European Journal of Operational Research*, 2000.

[11] Ki Young Kim. *Mathematical Models and Search Techniques for Containership Load Planning*. PhD thesis, Pusan National University, 1998.

[12] Ki Young Kim and Kap Hwan Kim. A routing algorithm for a single straddle carrier to load export containers onto a containership. *International Journal of Production Economics*, 1999.

[13] Erhan Kozan and Peter Preston. Genetic algorithms to schedule container transfers at multimodal terminals. *International Transactions in Operational Research*, 1999.

[14] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Inc., 1995.

[15] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, Inc, 1998.

[16] Peter Preston and Erhan Kozan. An approach to determine storage locations of containers at seaport terminals. *Computers and Operations Research*, 2001.