Solution techniques for a crane sequencing problem

Jin Shang
West Virginia University

SOLUTION TECHNIQUES FOR A CRANE SEQUENCING PROBLEM

Jin Shang

Dissertation Submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Industrial Engineering

Department of Industrial & Management Systems Engineering

Morgantown, West Virginia
2006

Keywords: Crane sequencing problem, Tabu search, Simulated annealing, Meta-heuristics

ABSTRACT

SOLUTION TECHNIQUES FOR A CRANE SEQUENCING PROBLEM

Jin Shang

In shipyards and power plants, relocating resources (items) from existing positions to newly assigned locations are costly and may represent a significant portion of the overall project budget. Since the crane is the most popular material handling equipment for relocating bulky items, it is essential to develop a good crane route to ensure efficient utilization and lower cost. In this research, minimizing the total travel and loading/unloading costs for the crane to relocate resources in multiple time periods is defined as the crane sequencing problem (CSP). In other words, the objective of the CSP is to find routes such that the cost of crane travel and resource loading/unloading is minimized. However, the CSP considers the capacities of locations and intermediate drops (i.e., preemptions) during a multiple period planning horizon. Therefore, the CSP is a unique problem with many applications and is computationally intractable. A mathematical model is developed to obtain optimal solutions for small size problems. Since large size CSPs are computationally intractable, construction algorithms as well as improvement heuristics (e.g., simulated annealing, hybrid ant systems and tabu search heuristics) are proposed to solve the CSPs. Two sets of test problems with different problem sizes are generated to test the proposed heuristics. In other words, extensive computational experiments are conducted to evaluate the performances of the proposed heuristics.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my committee chair and advisor, Dr. Alan McKendall, who spent numerous hours to help me finish this dissertation. He always gave me valuable guidance, comments and reviews. I feel greatly honored and privileged to have studied under the tutelage of Dr. Alan McKendall. His patience, guidance, valuable suggestion and incalculable support have been crucial ingredients in bring this dissertation to completion. Without his help, I could not have finished this research.

I am also grateful to my other committee members, Dr. Wafik H. Iskander, Dr. Majid Jaraiedi, Dr. Hong-Jian Lai and Dr. Elaine Eschen for their helpful comments and suggestions during this research.

Also, I would like to thank my wife, Yan Wang for her love, unconditional support, and never-ending patience.

Thanks to all my friends for their support and encouragement.

Special thanks to the department of Industrial and Management Systems Engineering, West Virginia University, for providing me with good conditions for learning and conducting research.

Finally, I extend my deepest thanks to my parents for always arranging conditions for me to become successful.

TABLE OF CONTENTS

LIST OF TABLES

LIST FIGURES

CHAPTER 1

INTRODUCTION

1.1 Routing Problems

During recent years, logistics and transportation systems have become increasingly complex. This development is partly due to the large number of company mergers which leave logistics planners with larger and more complex problems. Another issue complicating logistic systems is the increased focus on timeliness. Time has become an extremely valuable resource. Nowadays most logistics systems must operate under time constraints. Furthermore, logistics and transportation account for a large portion of the economies in developed countries. Due to the rapid increase in fuel cost, moving commodities from suppliers to customers is a costly operation and may represent a significant portion of the overall budget. Therefore, companies should focus their attention on developing systems that could aid logistics managers to lower costs and to achieve greater efficiency.

A lot of research has been performed in the field of logistics, and many problems have already been defined, from the well-known traveling salesman problem to complex dynamic routing problems. Generally, routing problems are responsible for determining how resources (items) are delivered from original locations to destination locations or from supplier(s) to customers. Routing problems that involve the periodic collection and delivery of goods and services are of great practical importance. Common examples of such problems include mail delivery, newspaper delivery, parcel pickup and delivery, trash collection, school bus routing, snow removal, inventory rearrangement and fuel oil delivery. The most practical objectives for such problems are cost minimization and service improvement (**Frederickson *et al.,* 1978**). For instance, the objective may be to minimize the transportation cost, construction cost, distance, travel time, inventory cost and environmental concerns; or maximize safety, demand satisfaction, accessibility, quality, flexibility and reliability of service, facility utilization, profit, and economic development (**Current, 1993**). These objectives usually can be measured in distance, travel time or cost.

1.2 Vehicle Routing and Traveling Salesman Problems

A classical problem in distribution logistics is the optimization of the routes of a set of vehicles of given capacities that must deliver goods to a set of customers on a transportation network, starting from and returning to a common depot. This problem is known as the vehicle routing problem (VRP). The VRP is an important problem and can be applied to a wide range of logistics systems. It was initially studied by **Dantzig and Ramser (1959)** and **Clarke and Wright (1964).** A lot of effort has been devoted to research on various aspects of the VRP. Another important routing problem, the traveling salesman problem (TSP), considers the route for a single vehicle (or a traveling salesman), which is a special case of the VRP (single vehicle routing problem). It may be defined as follows. Given a set of customers and the cost of travel between all pairs of customers, the problem is to determine the least-cost salesman route which visits each customer exactly once (**Burkard *et al.,* 1998**).

Extensive research on the VRP and the TSP has led to the development of continuously improving exact solution methods (**Groetschel and Holland, 1991)**. However, exact methods are still unable to handle large-scale problems within reasonable computational time, and efficient heuristics are needed to solve such problems. A literature review of the VRP and the TSP will be given later in Chapter 3. Although the terms, “vehicle” and “traveling salesman” are used often in the routing problem literatures, many different kinds of material handling equipment (vehicles), such as trucks, tractors, and cranes are used to transport items from initial locations to destination locations.

1.3 Material Handling Equipment (Vehicles)

As previously mentioned, many kinds of material handling equipment or vehicles are widely considered in routing problems. To operate and maintain, they usually require a large portion of the project’s budget. The equipment is used to move the resources (e.g., construction materials, earth, petroleum products, or heavy materials) from their original locations to their destination locations. Typically, there are two cases of using material handling equipment. First, vehicles can be used to move items from suppliers (sources) to customers (destinations) over relatively long distances. In other words, suppliers and customers are different and are far apart. In the second, equipment may be used to move materials over relatively short distances. For example, material handling equipment is used to move items on or off vehicles (e.g., trucks, trains, ships, etc.) to locations within a warehouse. In other words, vehicles may be used to move materials either around or within a construction site, factory, or warehouse. In this research, the latter is considered.

Usually operators are needed to operate material handling equipment. Also, operators may control equipment by moving levers or foot pedals, operating switches, or turning dials. They may also set up and inspect equipment, make adjustments, and perform minor repairs when needed. Operators are classified by the type of vehicles or equipment they operate, such as trucks, industrial trucks, excavators, or cranes. In addition, each piece of equipment requires different skills to move different types of loads. There are four traditional kinds of material handling equipment: trucks, industrial trucks and tractors, excavation and loading machines, as well as cranes and towers (see **U.S. Department of Labor, 2004**). Below, the equipment types are discussed further as well as some of their advantages and disadvantages.

1) Trucks, ships, trains, and airplanes are used to transport items over relatively long distances. Trucks are the dominant mode of freight transportation in many countries because they are the most flexible. They are flexible because items can be delivered to almost any location in any continent. Ships have very high capacities and very low cost; but transit times are very slow, and large areas of the world are not directly accessible to ships. Trains encourage large shipments over long distances with low cost, but transit times are long and may be subjective to variability. Airplanes are very fast but may have much higher cost (**Chopra and Meindl, 2004**).

2) Industrial truck and tractor operators drive and control industrial trucks or tractors equipped with lifting devices, such as a forklift, boom, or trailer hitch. A typical industrial truck, often called a forklift or a lifting truck, has a hydraulic lifting mechanism and forks. Industrial truck operators use these forks to carry loads on a skid, or pallet, within or around a factory or warehouse over relative short distances. Industrial trucks may also have trailers loaded with materials, goods, or equipment within factories, warehouses or around outdoor storage areas.
3) Excavation and loading machines include bulldozers, excavators, backhoes, etc. A bulldozer is a powerful vehicle equipped with a blade and can be found on large and small scale construction sites, mines, roadsides, military bases, heavy industry factories, and large governmental projects. A backhoe is a piece of excavating equipment consisting of a digging bucket on the end of an articulated arm. A backhoe attached to a swiveling cab on top of tracks is called an excavator. Operators dig and load sand, gravel, earth, or similar materials into trucks or onto conveyors using machinery equipped with scoops, shovels, or buckets. Construction and mining industries employ virtually all excavation and loading machine operators.
4) Crane and tower operators lift materials, machinery, or other heavy objects. They extend or retract a horizontally mounted boom to lower or raise a hook attached to the load line. Most operators coordinate these maneuvers in response to hand signals and radioed instructions. Operators position the loads from the on-board

console or from a remote console at the site. While crane and tower operators are most noticeable at office buildings and other construction sites, the largest group works in primary metal and metal fabrication. Also, they are used to move materials/equipment at power plants, ship yards, etc. Cranes are most widely used to move heavy, large, bulky resources within or around a construction, manufacturing or warehouse sites. It is the material handling equipment considered in this research.

When many items need to be moved using a crane, the crane operator needs to decide which items should be moved first, second, third and so on, such that an objective is minimized (e.g., minimizing transportation cost as well as loading and unloading cost). In this research, this problem is defined as the crane sequencing problem (CSP).

1.4 An Application of the CSP

Based on the above discussions, it is obvious that the CSP is closely related to the TSP. In addition, the CSP considered in this research was defined from a real world problem. More specifically, the sequence in which the overhead crane inside a reactor containment building at a nuclear power plant moves resources (toolboxes) from one location to another will be considered. This problem was defined in **McKendall *et al.* (2006)**. However, a complete explanation of how the CSP was developed is described below, but first, explanations of related problems, the resource constrained project scheduling problem (RCPSP) and the dynamic space allocation problem (DSAP) are needed.

At electric power plants, outage activities (e.g., laydown, preventative maintenance, and surveillance activities) are scheduled by solving the RCPSP. In other words, the RCPSP determines the start and finish times for each activity such that outage duration is minimized and constraints on resources, space, and precedence relationships between activities are satisfied. The resources needed to perform outage activities are cranes, toolboxes, space and work crews (e.g., laborers, operators, engineers, etc). A detail investigation of the RCPSP can be found in **Kolisch and Hartmann (2004)**.

Once the outage activities are scheduled at a power plant, the locations of the activities and resources need to be determined such that the total distance the resources travel throughout the duration of the outage is minimized. This problem was defined in **McKendall *et al.* (2005)** as the dynamic space allocation problem (DSAP). More specifically, during certain time periods, some resources are required to perform activities, and other resources are idle. The DSAP assigns activities and their required resources to workspaces and idle resources to storage spaces with respect to minimizing the sum of the distances the resources travel. Therefore, the input data for this problem are the schedule of maintenance activities obtained from solving the RCPSP, the list of resources required to perform each activity, locations of available workspaces used to perform activities, locations of storage spaces used to store idle resources, capacities of the storage spaces, and the distances between locations. The outputs are the assignment of activities (required resources) to workspaces and idle resources to storage spaces for each period (or change in the schedule of activities), and the total distance the resources travel. For illustrative purposes, a DSAP instance is given below.

Consider a DSAP instance which considers 2 time periods ($T = 2$), 9 resources ($R = 9$), and 6 locations ($N = 6$). The input data are given in **Figures 1.1** and **1.2** as well as **Table 1.1**. The layout configuration in **Figure 1.1** shows that locations 1 to 3 and locations 4 to 6 are workspaces and storage spaces, respectively. Each location has a capacity of 3 resources. In **Figure 1.2**, the distances between the 6 locations are given and defined as $D = [d_{ij}]$, $i, j = 1, 2, \ldots, N$ where d_{ij} is the distance from location i to j. There are 5 activities in this DSAP instance, and their resource requirements and the periods they are performed are given in **Table 1.1**. For example, activities A1, A2 and A3 are performed in period 1, and A1 is also performed in period 2, along with activities A4 and A5. Activity A1 requires resource 1, and activity A5 requires resource 7. In period 1, resources 1, 2, 3 and 4 are used to perform activities. However, resources 5, 6, 7, 8 and 9 are idle. Therefore, the DSAP is used to assign the activities and their required resources to workspaces and the idle resources to storage spaces. Using the mathematical formulation for the DSAP presented in **McKendall *et al.* (2005)**, the optimal assignment is obtained and given in **Figure 1.3**. In period 1 ($t = 1$), activities A1, A2 and A3, along with their required resources, are assigned to locations 1, 2 and 3, respectively. In addition, idle resources (8, 9), (5, 6) and (7) are assigned to locations 4, 5 and 6 (i.e., storage spaces 1, 2 and 3), respectively. The resources in bold represent resources which are to be relocated. That is, resources 2 and 3 are moved from location 2 to location 5 at the beginning of period 2. Thus, the total distance the 2 resources moved is 2 distance units. Also, resources 5 and 6 are moved from location 5 to location 2 at the beginning of period 2, which gives a travel cost of 2 distance units for resources 5 and 6. In addition, resources 4 and 7 are reassigned to locations 6 and 3, respectively, which gives a travel cost of 2 distance units. Therefore, the total distance the resources travel is 6 distance units.

In the above DSAP instance, an overhead crane is used to move resources 2 – 7 from their initial locations to their destination locations at the beginning of period 2. However, the DSAP does not specify the order in which the resources are moved. The order in which the resources are moved with respect to an objective is defined as the CSP. The objective considered in **McKendall *et al.*** (2006) was minimizing the distance the crane traveled. However, in this research, the objective of minimizing the cost of loading and unloading the crane is also considered. Therefore, the input data for the CSP are the assignments of activities (and their required resources) and idle resources to work spaces and storage spaces, respectively (i.e., DSAP solution). Other input data for the CSP will be discussed in the next Chapter, and a formal definition will be given.

Work space 1 (Location 1)	Work space 2 (Location 2)	Work space 3 (Location 3)
Storage space 1 (Location 4)	Storage space 2 (Location 5)	Storage space 3 (Location 6)

Figure 1.1: Layout configuration for the DSAP instance.

i/*j*	1	2	3	4	5	6
1	0	1	2	1	2	3
2	1	0	1	2	1	2
3	2	1	0	3	2	1
4	1	2	3	0	1	2
5	2	1	2	1	0	1
6	3	2	1	2	1	0

Figure 1.2: Distance matrix D for the DSAP instance.

Period	Activity	Required Resources	Idle Resources
1	A1	1	5, 6, 7, 8, 9
	A2	2, 3	
	A3	4	
2	A1	1	2, 3, 4, 8, 9
	A4	5, 6	
	A5	7	

Table 1.1: Activities and Resources Data for the DSAP instance.

t = 1

1 (*A1*)	**2,3** (*A2*)	**4** (*A3*)
8,9	**5,6**	**7**

t = 2

1 (*A1*)	5,6 (*A4*)	7 (*A5*)
8,9	2,3	4

Figure 1.3: Optimal Solution for the DSAP instance.

1.5 Other Applications of the CSP

Beside the application of the CSP in the power plant environment mentioned in section 1.4, there are other potential applications for applying the CSP in the areas of logistics or warehouse management. One application of the CSP occurs in the context of warehouse rearrangement. Since the demands of products change, warehouse managers may rearrange the layout of their products in warehouses. For example, products with high demand usually are located close to the input/output locations. Therefore, the CSP can be used to determine the most efficient way to rearrange the items in a warehouse. Another application of the CSP is in ship yards. When a ship arrives at the terminal, containers are normally discharged from the ship, mounted onto trucks by quay cranes, and then unloaded at various locations in the yard for storage. Before a ship arrives, terminal planners need to determine the sequence of discharging containers from the ship. The objective is to minimize the total loading and unloading as well as crane travel cost. Hence, there are several areas where the CSP can be applied.

CHAPTER 2

PROBLEM STATEMENT

2.1 Introduction

Once resource assignments are determined (i.e., a solution to the DSAP is obtained) in a power plant, warehouse, or shipyard environment, the CSP could be used to determine the order in which the material handling equipment moves the resources to their required locations. Various vehicles or material handling equipment may be used to relocate resources (items). Different types of resources require different handling and moving techniques. For example, forklifts may be used to move products within manufacturing plants and warehouses. In this research, a single polar or overhead crane with single capacity is considered to relocate resources, since these cranes are the most popular material handling equipment for moving large bulky resources within facilities. In addition, an operator is needed to operate the crane, and additional workers are needed to load and unload it. Therefore, the objective of the CSP is to determine the order in which the crane moves resources such that the sum of the material handling and the loading/unloading costs are minimized.

Often times, when the crane is needed to move many resources, bottlenecks occur. For example, in a construction or power plant environment, a number of activities performed during certain time periods may require multiple resources which are moved by the crane. In other words, when some activities are completed, other activities will start, and the resources required to perform these activities are loaded and moved to their desired locations by the crane. To reduce bottlenecks (e.g., total time and cost spent operating the crane), the objective of minimizing the sum of material handling and loading/unloading costs is necessary; thus, overall project cost is reduced. Therefore, it is essential to develop a good crane schedule to obtain the above objectives.

2.2 Problem Definition

The CSP was first defined in **McKendall *et al.* (2006)**, which is the problem of determining the sequences in which a single overhead crane moves resources to their assigned locations with respect to minimizing the total distance the crane travels. In other words, the objective is to find sequences (or routes in which resources are moved) for a crane that minimize the total distance traveled by the crane. However, in this research, the objective is to find routes for a crane such that the crane travel as well as loading/unloading costs are minimized. Therefore, loading/unloading costs have been added to the CSP. More specifically, when resources are rearranged, the crane is used to relocate the resources. First, a resource is loaded onto the crane. Second, the resource is moved from its original location to its new location. Third, the resource is unloaded. Then the process is repeated for each resource requiring a new location. Hence, the CSP determines the order in which the resources should be moved such that total cost is minimized. The total cost consists of two major costs. First, loading/unloading cost is the cost of loading the resources onto the crane and the cost of unloading the resources. Second, the crane travel cost is the cost of moving the resources from their original locations to their destination locations using the crane. Therefore, the output of the CSP is a sequence of resources, which are moved by the crane at each time when resource assignments are changed.

2.3 Problem Assumptions

There are several assumptions which make the CSP different from other related problems given in the literature. These assumptions are listed as follows and the major assumptions will be explained in detail below.

1) The CSP assumes that the locations of resources assigned at multiple periods are known in advance (DSAP solution is given); The initial locations of resources are known at first time period;
2) Capacity for a single overhead crane is one unit of a resource (or item);
3) At the first time period, the initial position of the crane is at location 1;
4) After the first time period, the initial position of the crane is the last position of the crane in the previous time period;
5) A temporary storage space may be necessary when the destination location for a resource to be moved is not available because of capacity restrictions. In addition, in some situations a temporary storage space may be required in order to reduce the travel cost;
6) It is assumed that any available location can be used as a temporary storage

space;

7) The resources are fixed at their locations if they are not scheduled to be moved at a specific time period;
8) Last, the objective of the CSP is to find the sequences in which resources are to be moved by the crane such that the sum of the loading/unloading and crane travel costs is minimized.

The CSP is a problem which considers multiple time periods (i.e., items may be reassigned to locations in different periods). As a result, the CSP determines the crane routes for each period items are rearranged. Without loss of generality, the initial location of the crane at the beginning of each period, not including period 1, is the last position of the crane in the previous period as defined in Assumption (3). However, if the initial position of the crane at the beginning of each period is known (i.e., independent of the previous period), the problem can be solved as a series of static CSPs, which is much easier than the CSP defined in this research. It is important to note that all items may not be rearranged from one period to the next as stated in Assumption (7).

Next, some of the above assumptions are explained in details.

2.3.1 Static vs. Dynamic

The CSP is a problem of determining routes (or sequences) for a crane when relocating resources in multiple time periods. In the literature, many problems have considered the single period (i.e., static environment) problem which means that the problem data are assumed to be static during the planning horizon. For example, if the initial location of the crane is known a priori for all time periods, then the CSP is static. In other words, the CSP for each time period can be obtained independently of the other periods. However, only the initial location of the crane (i.e., the location of the crane at period 1) is known. After the sequence of the crane is determined for the initial period, the initial location of the crane for the next period is the location of the crane at the end of the previous period. Hence, the CSP for each time period are dependent on the CSP for preceding and succeeding periods. Thus, the CSP is dynamic and should not be solved independently for each time period in order to obtain more efficient solutions.

2.3.2 Objective of the CSP

For the crane considered in the CSP, the status of the crane is either loaded or not loaded. Therefore, there are two corresponding types of moves: empty moves and non-empty moves (see **Figure 2.1**). An empty move is a move where the crane does not carry a resource (i.e., crane is not loaded); whereas a non-empty move is a move where the crane carries a resource (i.e., crane is loaded). An empty move may be necessary for the crane to obtain a resource to be moved. For instance, if there is no resource to be moved at the current location of the crane, the crane needs to perform an empty move to arrive at a location of a resource which needs to be moved.

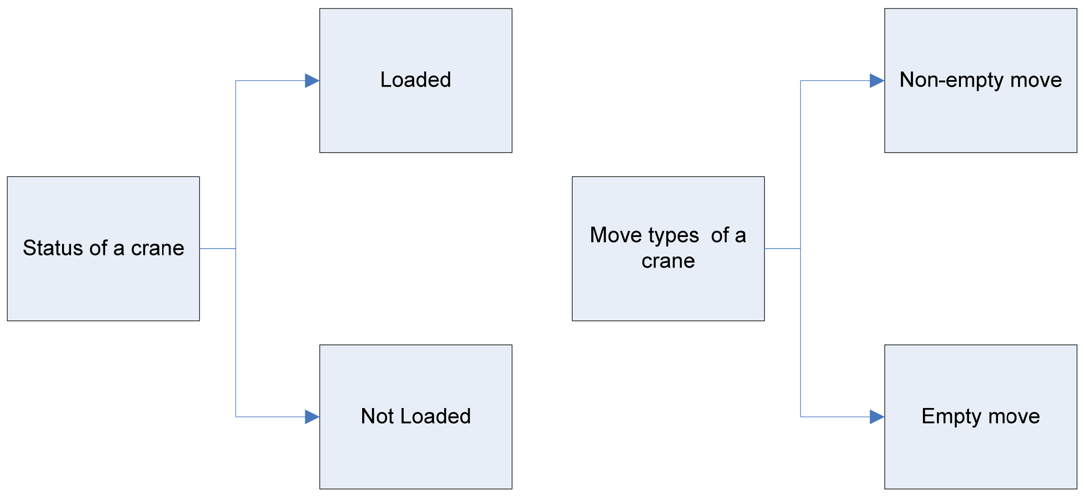

Figure 2.1: Status of crane and move types.

Based on the status or move types of the crane described above, the total costs of the CSP should include the costs related to empty moves and non-empty moves of the crane and the cost related with loading/unloading resources for the crane (see Figure 2.2). The distances the crane travels can be used as a criterion to measure costs when the crane moves empty or non-empty. As a result, the travel cost of the crane is the product of the distances the crane travels (empty or non-empty) and the cost per distance unit. In contrast, the loading/unloading cost is directly related to the total number of non-empty crane moves. That is, the loading/unloading cost is the product of the number of non-empty moves and the cost per non-empty move. In other words, the cost of a non-empty move is the cost of loading an item onto the crane and the cost of unloading the item once it reaches its destination location. This is called the loading/unloading cost (non-empty move cost). An example is given in section 2.4 to illustrate how the different costs are obtained.

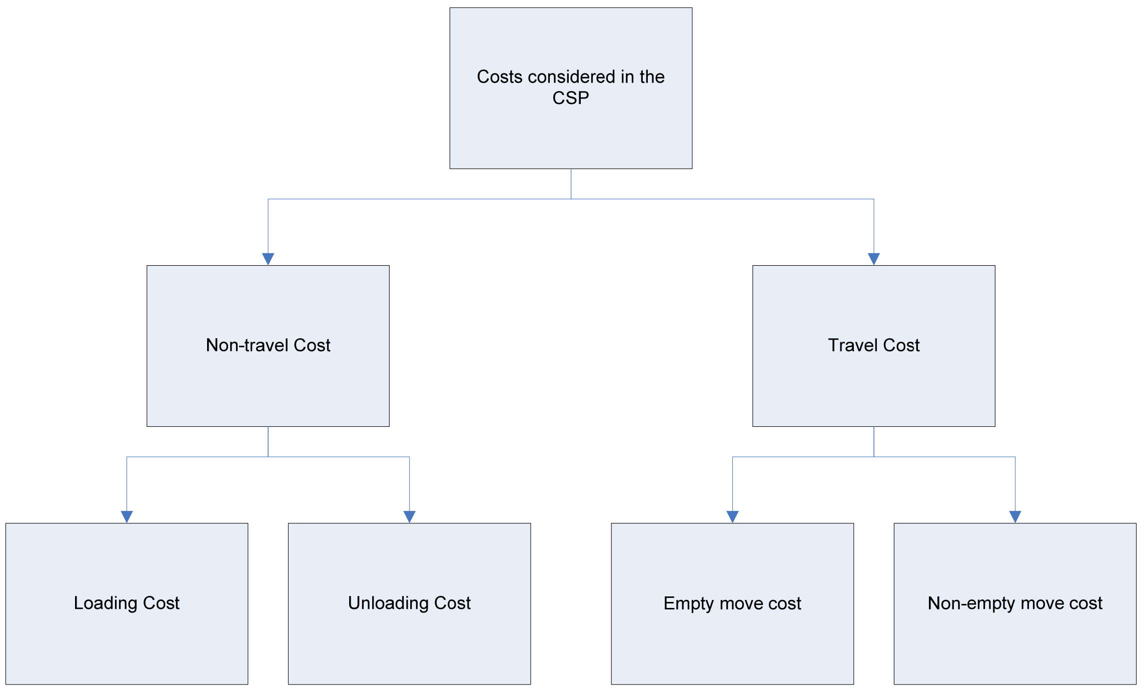

Figure 2.2: Costs considered in the CSP.

2.3.3 Temporary Storage Space

One of the most unique features of the CSP is the use of temporary storage spaces. As previously mentioned, temporary storage space may be necessary when:

1) The destination location of a resource currently being moved is at full capacity. In this case, either the resource has to be moved to another available location (called the temporary storage space) or the resource is moved at a later time when the destination location is available.

2) Routes of the crane can be improved (i.e., sum of total travel and loading/unloading costs for the crane is reduced) if temporary storage space is used.

Once a resource is moved to a temporary storage space, it is assumed that it is stored there until its assigned location is available and either i) the crane is at its temporary location or ii) after all the other resources have been moved to their assigned locations.

Note, in similar applications, when a resource can be put into an intermediate (temporary) location before it is delivered to its final destination location, the resource is preempted from moving to its final destination. In the literature, this is called “preemption” or “intermediate drop.” In this research, temporary locations are defined as follows:

a) Storage spaces not at full capacities

b) Workspaces that are currently available (no activity is currently being performed in those locations).

Below, a CSP instance is given to explain the temporary storage space assumptions as well as other assumptions for the CSP.

2.4 A CSP Instance

To illustrate the CSP and its assumptions, consider the CSP instance where the layout configuration and distance matrix are shown in **Figures 1.1** and **1.2**, respectively. Also, the assignment of activities (and their required resources) to workspaces and the assignment of the idle resources to storage spaces are given in **Figure 2.3**. This CSP instance has 3 time periods (T = 3), 9 resources (R = 9), 6 locations (L = 6) and 4 activities (A = 4). Also, each location has a capacity of 3 items.

In period 1, activities 1 and 2 are performed in locations 1 and 2 (or workspaces 1 and 2), respectively. Since resources (4) and (2, 3) are used to perform activities 1 and 2, they are also assigned to locations 1 and 2, respectively. In addition, idle resources (1, 8, 9), (5, 6), and (7) are assigned to locations 4, 5, and 6 (or storage locations 1, 2, and 3), respectively. Notice that activity 2 is performed in both periods 1 and 2, and activity 3 is performed in period 2. All the items in bold font represent the items which are to be relocated at the beginning of periods 2 and 3. More specifically, items 1 and 4 are reassigned to locations 1 and 4 at the beginning of period 2. In addition, items (1), (2, 3), and (5, 6) are reassigned to locations 6, 5, and 2, respectively, at the beginning of period 3. Therefore, the CSP determines the order in which the crane moves the items for each of the periods such that the sum of the travel and loading/unloading costs is minimized.

$t = 1$

4 (*A1*)	2,3 (*A2*)	
1,8,9	5,6	7

$t = 2$

1 (*A3*)	**2,3** (*A2*)	
4,8,9	**5,6**	7

$t = 3$

	5,6 (*A4*)	
4,8,9	2,3	1,7

Figure 2.3: Resource Assignments for the CSP instance.

Period	Sequence of resources moved by the crane	Actual moves of the crane	Distance units the crane travels	Number of resource moves	Cost
***t* = 2**	**1 - 4**	**1 - 4 - 1 - 4**	**3**	**2**	**$21**
t = 2	4 - 1	1 - 5 - 4 - 1 - 5 - 4	7	3	$44

Table 2.1: The CSP instance solutions at period $t = 2$.

As mentioned earlier, there are multiple periods (i.e., $T = 3$) considered in this CSP instance. As a result, the CSP will be used to generate $T - 1$ crane routes (i.e., crane routes for periods 2 and 3). Also, the capacity of the crane is one unit of an item. Assume that the travel cost per distance unit is \$5, and the cost of loading and unloading each item (i.e., each non-empty move) is \$3. In addition, the initial location of the crane at the beginning of period 2 is location 1 for this specific instance. At the beginning of period 2, the crane needs to move items 1 and 4 to locations 1 and 4, respectively. Hence, there are two possible sequences for the crane to relocate items 1 and 4 as shown in **Table 2.1**. If the sequence of items to be moved is 1–4, the crane first needs to perform an empty move from its initial location (i.e., location 1) to the location of item 1 (i.e., location 4) which gives a travel cost of 1 distance unit. Once the crane arrives at location 4, item 1 is loaded and prepared for moving. Next, the crane moves item 1 from location 4 to location 1 which gives a travel cost of 1 distance unit. Also, item 1 is unloaded. Therefore, loading/unloading cost for item 1 is \$3. Next, item 4 is considered to be moved, since it is the next item in the sequence. Since the crane is currently at location 1 where item 4 is located, item 4 is loaded and prepared for moving. Then, the crane moves item 4 from location 1 to location 4, which gives a travel cost of 1 distance unit, and it is unloaded. As a result, loading/unloading cost of item 4 is \$3. Therefore, for this sequence, the total loading/unloading cost is \$6, and the total crane travel cost is \$15 (i.e., \$5(3) where the total distance the crane travels is 3 distance units). Hence, the total cost of the sequence 1-4 is \$21. On the other hand, if the sequence of items to be moved is

4–1, the number of item moves is 3 and the crane travels 7 distance units. Therefore, the total cost of this sequence is \$44 (i.e., \$3(3) + \$5(7)). Note that for this sequence, the crane starts at location 1 where item 4 is loaded and moved from location 1 to a temporary storage location 5, since its destination location 4 is at full capacity. This is case (1) of temporary storage space usage. Initially, locations 1, 3, 5, and 6 can be used as temporary storage locations, since either no activity is currently being performed at these locations (workspace locations 1 and 3) or the locations (storage locations 5 and 6) are not at full capacities. However, since location 5 is closest to item 4's destination location (location 4), location 5 is selected as the temporary storage location. This specific move gives a loading/unloading cost of \$3 and a crane travel cost of \$10. Then the crane moves from location 5 to location 4 (crane travel cost is \$5), item 1 is loaded and moved from location 4 to location 1, and unloaded (crane travel cost = \$5 and loading/unloading cost = \$3). Now crane moves from location 1 to location 5 (crane travel cost = \$10), item 4 is loaded and moved to location 4, and unloaded (crane travel cost = \$5 and loading/unloading cost = \$3). If we consider the combined problems of the DSAP and the CSP, it may be better to eliminate the movement of item 4 from location 5 to location 4. But we are restricting ourselves here to the result of the DSAP, which may restrict the CSP solution space. However, combining the DSAP and the CSP can be considered in future research. Thus, the sequence 4-1, gives the crane route 1-5-4-1-5-4 indicating the sequence in which the locations are visited by the crane. Nevertheless, sequence 1-4, which cost \$21, is much better than sequence 4-1, which cost \$44.

Period	Sequence of resources moved by the crane	Actual moves of the crane	Distance units the crane travels	Number of resource moves	Cost
t = 3	1 - 2- 5 - 3 - 6	4 - 1 - 6 - 2 - 5 - 2 - 5 - 6	10	5	$65
***t* = 3**	**1 - 2- 5 - 3 - 6**	**4 - 1 - 2 - 5 -2 - 5 - 2 - 6**	**8**	**6**	**$58**

Table 2.2: The CSP instance solutions at period *t* = 3.

For period 3, the initial location of the crane is the location of the last position of the crane in period 2. As a result, the crane is located at location 4. Also, the number of possible resource sequences is 5! = 120. However, if a sequence requires temporary storage space to either reduce total cost or capacity restrictions of destination locations, then more than one crane route may be constructed from this sequence of resources. In **Table 2.2**, one of the crane sequences is considered to illustrate temporary storage location issues, specifically, how different crane routes are formed from the same sequence. Consider crane sequence 1–2–5–3–6. First, the crane moves empty from location 4 to location 1 (crane travel cost = $5). Second, item 1 is loaded and moved from location 1 to location 2 (crane travel cost = $5 and loading/unloading cost = $3), since using location 2 as temporary storage space reduces total cost, which considers case (2) above. Next, items 2, 5, 3 and 6 are loaded and moved in order from locations 2, 4, 2 and 4 to their assigned locations 4, 2, 4, and 2, respectively (crane travel cost = $20 and loading/unloading cost = $12). Because all other items had been moved to their destination locations, item 1 is loaded and moved from its temporary location (location 2) to its required location (location 6) and is unloaded (crane travel cost = $10 and loading/unloading cost = $3). Therefore, the cost of sequence 1-2-5-3-6, specifically route 4-1-2-5-2-5-2-6, is $58.

Note, the sequence indicates the order in which the items are moved and does not indicate when an item is moved from temporary storage space. However, the crane route indicates the order in which the locations are visited, which gives a more detailed solution. For the same sequence, sequence 1-2-5-3-6, specifically route 4-1-6-2-5-2-5-6, if the crane moves item 1 directly to its destination location, location 6, and then moves items 2, 3, 5 and 6 to their destination locations, the total cost is \$65. Therefore, using temporary storage location can reduce total cost.

Period	Sequence of resources moved by the crane	Actual moves of the crane	Distance units the crane travels	Number of resource moves	Cost
$t = 2$	1 - 4	1 - 4 - 1 - 4	3	2	\$21
$t = 3$	1 - 2- 5 - 3 - 6	4 - 1 - 2 - 5 -2 - 5 - 2 - 6	8	6	\$58
					\$79

Table 2.3: Optimal Solution for the CSP instance.

In **Table 2.3**, the optimal solution for this CSP instance is given as the following sequences for periods 2 and 3: 1–4 and 1–2–5–3–6, respectively. The total cost of this solution is \$79 = \$21 + \$58. The optimal solution was obtained using the binary integer linear program presented in this research for the CSP and CPLEX, Version 6.6. It took 0.1 minutes on a Pentium IV 2.8 GHz PC. However, for another small CSP with 9 items and 4 periods, it required 8.1 hours of computation time. Therefore, only small-size CSPs were solved in reasonable computation time using the mathematical model and CPLEX. As a result, several meta-heuristics were developed to solve large-size problems in reasonable time.

2.5 Research Objectives

The objectives for this research are listed below:

- Formulate a mathematical model for the CSP;
- Use the mathematical model to obtain optimal solutions for small-size CSPs;
- Develop construction algorithms to obtain initial solutions for the CSP;
- Develop a simulated annealing, hybrid ant system and tabu search heuristics to solve large-size CSPs;
- Generate a set of test problems to compare the solution techniques;

CHAPTER 3

LITERATURE REVIEW

3.1 Introduction

The CSP considered in this research is to find the sequence for a crane to move resources to their destination locations at the beginning of each time period. Many types of sequencing problems exist in the literature, which are widely researched. For instance, **Faggioli *et al.* (1998)** studied a cutting stock problem dealing with the generation of a set of cutting patterns that minimizes waste. **Smith *et al.* (1996)** considered a problem of optimally sequencing different car models along an assembly line according to contiguity constraints, while ensuring that the demands for each of the models are satisfied. **Wen *et al.* (1997)** compared the solution procedures of the flow shop sequencing problems that have been studied in the literature based on a tabu search heuristic. Generally, these sequencing problems belong to a set of problems called permutation problems (i.e., solutions of these problems can be represented as permutations of jobs or items). However, the CSP considered in this research is a much more complex and more general sequencing problem with special constraints, which have been discussed in Chapter 2. That is, first, the resources moved in the CSP are not identical and each resource has specific origin and destination locations. Second, temporary storage spaces may be used to store resources temporarily. In other words, resources may be dropped into temporary storage spaces, and picked up later. Third, multiple periods (i.e., multiple resource assignments) are considered in the CSP. For each period (after the first time period), the orders of resources to be moved by the crane should be determined. After the first time period, the initial position of the crane is the last position of the crane in the previous time period. In addition, a limited capacity is assumed for each of the locations. All of these assumptions make the CSP unique and much more difficult to solve. A survey of the literature reveals that the CSP is related to the TSP with additional assumptions such as non-Hamiltonian tours or preemption conditions. To illustrate the similarities and differences of the CSP with the related variations of the TSP in the literature, the VRP, which is a more general problem of the TSP, is discussed first.

3.2 Vehicle Routing Problem

The VRP is one of the prominent routing problems in the logistics area. It can be briefly described as a set of clients or customers with known and deterministic demands, which have to be served from a central depot or origin, with a fleet of delivery vehicles of known capacity. The total customer demand of a route must not exceed the vehicle capacity. Normally, the objective is to minimize the total distance traveled by the vehicle fleet, but it is also common to minimize total transportation (routing) costs. This combinatorial problem is *NP*-Hard (see **Garey and Johnson, 1979**). A VRP example is given in **Figure 3.1** with 9 customers and 3 vehicles. Three vehicles start from a Depot 0. Then 9 customers are assigned to one of three vehicles.

Each vehicle will visit the customers assigned to it exactly once and return to the depot. The objective is to minimize the total distance traveled by these vehicles (or total transportation cost).

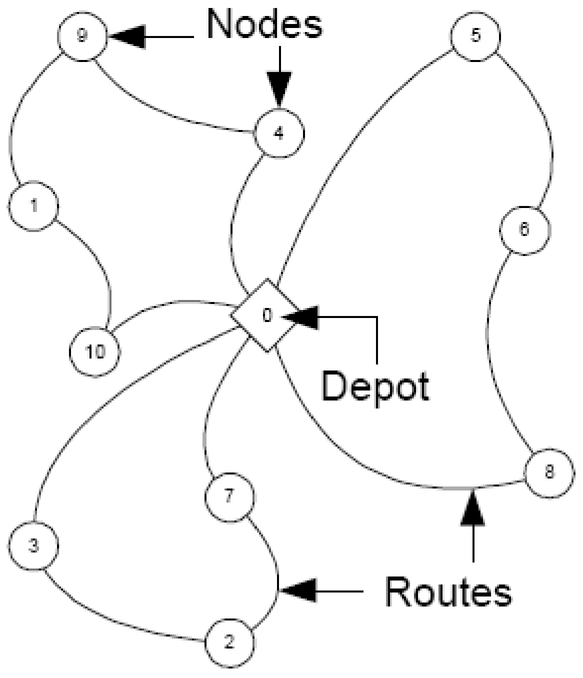

Figure 3.1: A VRP instance with 9 customers and 3 vehicles.

The VRP is an important sub-problem in a wide range of distribution systems and a lot of effort has been devoted to researching various aspects of the VRP. In the literature, starting from the basic version of the VRP, many variations have been considered, such as capacitated vehicle routing with pickups and deliveries, VRP with precedence constraints, open VRP, VRP with backhauls, and many others. A comprehensive and detailed study of the VRP and its variations can be found in **Bramel and Simchi-Levi (1999)**, **Crainic and Laporte (1999)** and **Ball *et al.* (1995)**.

Since the CSP determines the routes for a single crane to move resources when resource arrangements are changed, the single vehicle version of the VRP with pickup and delivery is paid more attention in this literature review. The single vehicle version of the VRP, which is a special case of the VRP, was defined as a TSP and will be discussed below.

3.3 Traveling Salesman Problems

If only a single vehicle is available for deliveries and the vehicle has enough capacity, this problem is known as the traveling salesman problem (TSP). The TSP belongs to the most basic, important, and investigated problems in combinatorial optimization and is *NP*-hard (**Burkard *et al.*, 1998**). **Figure 3.2** shows a depot (a black dot with a circle) and cities (black dots) that need to be visited by a salesman (or a vehicle). The salesman (or vehicle) is required to start from the depot, visit all the cities exactly once and return to the depot. The objective of the TSP is to minimize the total distance traveled by the vehicle (or salesman). A feasible solution for this TSP example is shown in **Figure 3.3**.

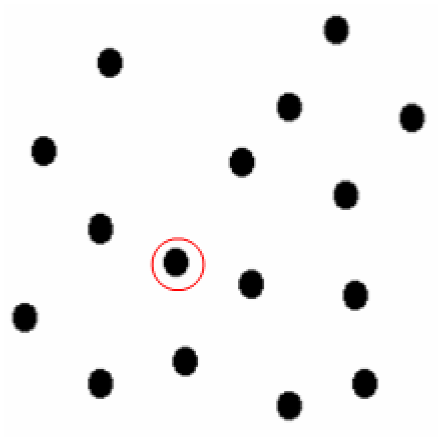
Figure 3.2: Cities and depot in a TSP instance.

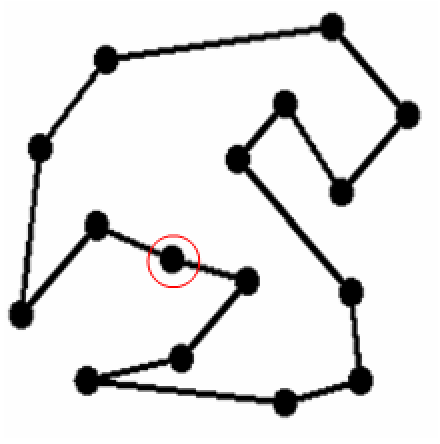

Figure 3.3: A feasible solution for the TSP instance.

Similar to the VRP, there are lots of variations of the TSP which exist in the literature. In this research, eight different factors are considered to categorize these variations of the TSP and the CSP. These factors are listed and discussed below.

1. # of Origins-Destinations ("*one to one*", "*one to many*", "*many to many*")
2. Number of product types (1, *m*)
3. Number of products per type (1, *k*)
4. Pickup/Delivery quantity at a location (1, *w*)
5. Capacity of a vehicle (1, *Q*, ∞)
6. Hamiltonian Tour ("*yes*", "*no*")
7. Preemption ("*yes*", "*no*")
8. Capacity of a location (1, *N*, ∞)

The factor, # of Origins-Destinations, considers the relation between the original location(s) and destination locations for items: "one to one" if each item picked up from an original location has a specific destination (or delivery location); and "one to many" if there is a single original location (depot) and multiple delivery locations (destination locations) for items, and items picked up from the original location (or depot) can be delivered to many destination locations; “many to many” if there are multiple pickup locations (original locations) and multiple delivery locations (destination locations) for items, and items picked up from any original location can be delivered to any one of the destination locations;

Number of item types and number of items per type are the number of item types delivered by the vehicle and number of items for a single item type. The number of item types can be either 1 or m. In addition, the number of items for a single item type can be either 1 or k.

The factor, pickup/delivery quantity (1, w) at a location, considers the amounts of products picked up or delivered at a location, which can be either a single unit or a batch size of w units. Also, the capacity of the single vehicle can be 1, Q, or ∞.

If the tour is a Hamiltonian tour (“*yes*”), then each location can be visited by the vehicle exact once. On the contrary, each location can be visited by the vehicle more than once which is a non-Hamiltonian tour (“*no*”). Obviously, non-Hamiltonian tours make the problem more complex.

Preemption is another important factor consider in the CSP. If preemption can occur (“*yes*”), an item (or resource) can be stored in a temporary storage space one or more times before it is moved (or delivered) to its destination location. Otherwise, if preemption is not allowed (“*no*”), no temporary storage is permitted and an item must be delivered to its destination location directly once it is picked up.

Moreover, the capacity of each location can be 1, N, or ∞. If a location is at full capacity, it is unavailable at that time. That is, no more items are permitted to be moved into this location before at least one item stored in this location is moved. Use "∞" when the capacity is not considered or capacity is unlimited.

Based on the factors discussed above, the CSP is a problem with factors "*one to one*" origins and destinations relation; it has "*m*" item (or resource) types; the number of items per type is "1"; a single unit of each item is picked up and delivered to a location; the capacity of the vehicle is "1"; the Hamiltonian tour is not satisfied (i.e. "*no*"); preemptions (or intermediate drops) are allowed; and capacity of a location is "*N*." Hence, the CSP can be represented using the format *one-one*/*m*/1/1/1/*no*/*yes*/*N*. As mentioned previously, several variations of the TSP known in literature are closely related to the CSP. These are: Q-delivery TSP, capacitated TSP with pickups and deliveries (CTSPPD), shortest route cut and fill problem (SRCFP), one-commodity pickup-and-delivery TSP (1-PDTSP), TSP with backhaul (TSPB), TSP with pick-up and delivery (TSPPD), swapping problem (SP), TSP with delivery and backhauls (TSPDB); pickup and delivery TSP (PDTSP), capacitated dial-a-ride problem (CDARP), stacker crane problem (SCP); motion planning problem (MPP); and warehouse rearrangement problem (WRP). These problems are summarized in **Table 3.1** according to their corresponding factors and will be discussed individually in the following sections.

Problem Name	# of Origins - Destinations	# of Product Types	# of Products per Type	Pickup/Delivery Quantity at a Location	Capacity of a Vehicle	Hamiltonian	Preemption	Capacity of a Location
TSP	*one-many*	1	k	1/*w*	∞	*yes*	*no*	∞
Q-delivery TSP	*many-many*	1	k	1	Q	*no*	*yes*	∞
CTSPPD/SRCFP	*many-many*	1	k	1	Q	*no*	*no*	∞
1-PDTSP	*many-many*	1	k	*w*	Q	*yes*	*no*	∞
SP	*many-many*	*m*	k	1	1	*no*	*yes*	∞
TSPB	*one-many*	2	k	1	∞	*yes*	*no*	∞
TSPPD/TSPDB	*one-many*	2	k	*w*	Q	*yes*	*no*	∞
PDTSP	*one-one*	*m*	1	1	∞	*yes*	*no*	∞
CDARP	*one-one*	*m*	1	*w*	Q	*no*	*yes*	∞
SCP/MPP	*one-one*	*m*	1	1	1	*no*	*yes*	∞
WRP	*one-one*	*m*	1	1	1	*no*	*yes*	1
CSP	*one-one*	*m*	1	1	1	*no*	*yes*	*N*

Table 3.1: Factors of Closely Related Problems to the CSP.

3.3.1 *Q*-delivery TSP

Chalasani and Motwani (1999) defined a *Q*-delivery TSP (*Q*-TSP) as: given a vehicle with a maximum capacity of *Q*, *N* identical products (i.e., single object type) in *N* arbitrary locations, and each of another *N* locations requires a product (i.e., "many to many" for factor "# of origins – destinations" and single object for pickup/delivery at a location); the objective is to find a shortest tour for the vehicle in which all the products can be delivered to their locations without exceeding the capacity of the vehicle. Since only a single unit of product will be picked or delivered at a location, the vehicle only needs to visit each location exact once. Therefore, Hamiltonian tour is satisfied. Also, the TSP is a special case of *Q*-delivery problem, since replacing each city of the TSP by a location and a product yields an instance of the 1-delivery TSP. In this instance, the vehicle has to simply find a shortest tour that visits all the locations (i.e., a TSP solution), since any product that is picked up by the vehicle can immediately be delivered to this location. In addition, the permitted preemptive case for *Q*-delivery TSP is considered in **Charikar *et al.* (2002)**, where products can be dropped at intermediate location and picked up to deliver later. This problem belongs to $\textit{many-many}/1/k/1/Q/\textit{yes}/\textit{yes}/\infty$.

3.3.2 Capacitated TSP with Pickups and Deliveries

The capacitated TSP with pickups and deliveries (CTSPPD) was discussed in **Anily and Bramel (1999).** CTSPPD consists of *N* pickup customers and *N* delivery customers and a vehicle with limited capacity of *Q*. One depot used as the starting and ending point for the vehicle. The vehicle picks up a unit of product from a pickup customer and deliveries a unit of product to any delivery customer (i.e., "many to many"). The objective is to determine a minimal length feasible tour that picks up and deliveries all products and does not violate the vehicle capacities of *Q* units. This problem is equivalent to the Q-delivery TSP. However, a relaxation of the assumption of unit size loads without loss of generality was also discussed by the authors. For instance, a delivery (or pickup) of size 5 units can be delivered (or picker up) in 2 parts, one for 2 units and later one for 3 units. This is when each delivery or pickup load is allowed to split in the context of inventory repositioning. Under this assumption, Hamiltonian tour is not necessary in this case. That is, multiple visits for a location will be tolerated. Based on theser factors, it is a *manye-many*/1/*k*/1/*Q*/*no*/*no*/∞TSP.

Henderson *et al* (2003) studied another generalization of the TSP under construction environment where the objective is to find a vehicle route that minimizes the total distance traveled by a single earthmoving vehicle between cut and fill locations. It is defined as a shortest route cut and fill problem (SRCFP), which is another case of the CTSPPD, since it consists of two location types, cut locations and fill locations. At each cut location, it loads a unit of earth, and travels to a fill location where it deposits the unit of earth. Therefore, the vehicle visits every unit cut and every unit fill locations exactly once. In other words, starting at a cut location, the vehicle visits cut and fill locations alternately and finally return to its starting location. This process is repeated until all excess earth has been moved. **Lim *et al.* (2004)** relaxed the Hamiltonian tour assumption. The vehicle can visit the same location more than once if there is more than one unit cut (or fill) at that location. They also extended SRCFP to include multiple vehicles and a makespan objective.

3.3.3 One-Commodity Pickup-and-Delivery TSP

One commodity Pickup-and-Delivery TSP (1-PDTSP) was developed in **Hernandez-Perez and Salazar-Gonzalez (2004).** In this problem, a capacitated vehicle starts and ends the route at the depot like the traditional TSP. However, the customers are partitioned into two groups: delivery customers and pickup customers. A single type of commodity (or product) is delivered from the depot and pickup customers to delivery customers by the capacitated vehicle. Products picked up from a pickup customer can be supplied to any delivery customer (i.e., many to many). Each delivery customer requires a given amount of the commodities, while each pickup customer provides a given amount of the commodities (i.e., multiple units of products at a pickup/delivery location). The object is to minimize the distance tour for the vehicle visiting each customer once (i.e., Hamiltonian tour is satisfied) and satisfying the customer requirements without violating the vehicle capacity. In addition, preemption is not allowed in this problem. Moreover, Q-delivery TSP is the special case of the 1-PDTSP as well, where delivery and pickup quantities are all equal to one unit (**Hernandez-Perez and Salazar-Gonzalez, 2004**). Since a single product type, a capacitated vehicle and the Hamiltonian tour, etc are considered, this problem is represented as $manye\text{-}many/1/k/w/Q/yes/no/\infty$.

3.3.4 Swapping Problem

The swapping problem (SP) was proposed in **Anily and Hassin (1992)**. For this problem, each location initially may contain a product of a known type. A final state, specifying the type of product desired at each location, is also given. A single vehicle of unit capacity is available for shipping products among the locations so that the requirements of all locations are satisfied. The SP is more general than the variations of the TSP discussed above since the SP consists of a number of different product types. Each location is associated with the type of product currently at the location (if any) and the desired product type (if any) (see **Anily and Bramel, 1999**). In addition, the set of product-type is partitioned into two sets: products that may be temporarily dropped at intermediate locations before reaching their destinations and products that have to be shipped directly from their origins to their destination locations. Hence, the preemption is allowed for those products, which can be temporarily stored intermediately. Moreover, multiple visiting a location is permitted for this problem since a product will be moved out and another type of product will be moved in at each location (i.e., Hamiltonian tour is not satisfied). The objective is to design a route that starts and ends at depot and requirements of all locations are satisfied so that the total distance is minimized (**Anily *et al.,* 1999**). Also, **Chalasani and Motwani (1999)** considered the special case of two product types which, in the context of the SP, is equivalent to the CTSPPD with $Q = 1$. The SP can be stated as *many-many*/*m*/*k*/1/1/*no*/*yes*/∞.

3.3.5 TSP with Backhaul

TSP with backhual (TSPB) is a TSP with precedence constraint, which considers two types of customers: delivery customers and backhaul customers. It is defined as an un-capacitated vehicle must visit all the delivery customers before visiting a backhaul customer (see **Gendreau *et al.,* 1996 and Gendreau *et al.,* 1997**). The backhaul customers are different from those pickup customers discussed in above sections, where the pickup customers may delivery products to any of the delivery customers including the depot, while backhaul customers are restricted to delivery products from their locations to the depot. Therefore, there are two kinds of products considered in this problem. One is delivered from the depot to delivery customers, and the other is picked up from backhaul customers and delivered to the depot. Therefore, for either type of these products, the factor # of origins – destinations is "one – many". Thus, when leaving the depot, the vehicle carries the total backhaul requirements and gets all the products picked up from backhaul customers when back to the depot. The objective of the TSPB is to determine a least-cost Hamiltonian tour such that all backhaul customers are visited after all delivery customers. Hence, it is a *one-many*/2/*k*/1/∞/*yes*/*no*/∞ TSP.

3.3.6 TSP with Pick-Up and Delivery

Mosheiov (1994) introduced a TSP with Pick-up and Delivery (TSPPD). There are also two kinds of customers: backhaul customer and delivery customer. Products collected from backhaul customers must be transported to the depot, similarly as the TSPB. However, a capacitated vehicle is considered in this problem and it starts and ends at a depot. At a delivery customer, the vehicle unloads required products from the depot; while at backhaul customer, the vehicle loads products that are to be delivered to the depot. Unlike the TSPB, there is no restriction that all delivery customers must be visited before any backhaul customer is visited. The objective of the TSPPD is to find a Hamiltonian tour with minimal distance traveled by the vehicle. **Anily and Mosheiov**

(1994) also discussed the TSPPD, but renamed as TSP with delivery and backhaul (TSPDB). Both of these problems can be represented as $one\text{-}many/2/k/w/Q/yes/no/\infty$.

3.3.7 Pickup and Delivery TSP

In this version of the TSP, customers are also of two types: pickup customers and delivery customers. However, each pickup customer is associated with one and only one delivery customer and a pickup customer should be visited before its associated delivery customer (i.e., a "one to one" of origins – destinations relation and multiple product types since products pickup up at each pickup customer will be delivered to their unique delivery customer or destination location). The un-capacitated vehicle starts and ends at the depot. The objective of problem is to find a Hamiltonian tour with minimal distance traveled by the vehicle such that each pickup customer is visited before its associated delivery customer (**Renaud *et al.*, 2002)**. The Pickup and delivery TSP (PDTSP) can be seen as a generalization of the TSPB (see **Renaud *et al.*, 2000)** since an un-capacitated vehicle is used here. Because each customer is only visited exactly once, Hamiltonian tour is satisfied. Moreover, preemption (i.e., temporary drop) is not allowed in this problem either. This problem belongs to $one\text{-}one/m/1/1/\infty/yes/no/\infty$.

3.3.8 Capacitated Dial-A-Ride Problem

The capacitated dial-a-ride problem (CDARP) is another TSP with precedence relations where a capacitated vehicle should transport a number of passengers (see

Miyamoto *et al.*, 2003). This problem involve dispatching a vehicle to satisfy demands form a set of customers who call a vehicle-operating agency requesting to be picked up from a specific location and delivery to a specific destination. The goal is to minimize the total distance traveled by the vehicle in transporting all the customers (**Hunsaker and Savelsbergh, 2002**). The Hamiltonian tour is not satisfied for this problem since customers may have same destination locations (i.e., multiple visits for a location is permitted). In addition, the preemption case of the CDARP was discussed in **Charikar and Raghavachari (1998)**. The PDTSP can be seen as a special case of the CDARP with an un-capacitated capacity vehicle and a Hamiltonian tour. In addition, this problem differs from the *Q*-delivery TSP in that each customer must be dropped off at a specific destination. Hence, this problem is $one\text{-}one/m/1/w/Q/no/yes/\infty$.

3.3.9 Stacker Crane Problem

Another related work is the stacker crane problem (SCP). **Frederickson *et al.* (1978)** first introduced this problem. This problem also involves making deliveries with a vehicle of unit capacity like the CDARP. The products considered are not identical and each product has a specific destination location (**Charikar *et al.*, 2002**). The goal is to find a shortest tour that performs the required transportation. **Frederickson and Guan (1992)** considered the preemptive case of this problem. In this case, products can be dropped, and picked up and transported at some later time. This SCP is a special case of the swapping problem since there is only one unit of each product type in the SCP (**Anily *et al.*, 1999)**. The SCP is also special case of the CDARP with a unit capacitated vehicle.

Based on the factors of the SCP (i.e., *one-one*/*m*/1/1/1/*no*/yes/∞), the SCP is one of most closely related problems to the CSP, which has factors of *one-one*/*m*/1/1/1/*no*/*yes*/*N*. However, there are still differences between them. First, the only reason to drop an object to temporary storage space in the SCP is to find a better rout with lower travel cost. However, in CSP, not only to lower travel cost, the another reason to use temporary storage space is the capacity of a location. If the destination location of a resource being carried is at full capacity, this resource has to be moved a temporary storage space. Also, multiple time periods (i.e., multiple resource assignments) are considered in the CSP. However, only 2 states (initial state and final state) are considered in the SCP. Therefore, the SCP is more general than the SCP, which makes the CSP much harder to solve and change the nature of the algorithm to solve the CSP from SCP.

3.3.10 Warehouse rearrangement problem

Another related problem found in the literature is warehouse rearrangement problem (WRP), which was defined in **Christofides and Colloff (1973).** This problem of rearranging items in a warehouse use a single vehicle of unit capacity such that the total cost of rearranging the items from one known configuration (initial/old arrangement) to another (new arrangement) is minimized. Similarly as the CSP, the location capacity is considered in this problem. During the rearrangement of items, a predetermined temporary storage space is used if maximum number of items (i.e., capacity) is met for a destination location. The output is the sequence of item movements required to achieve the new arrangement of items and the total rearrangement cost. If group this problem into the above variations of the TSPs, it belongs to a *one-one*/*m*/1/1/1*no*/*yes*/1. Unlike the CSP, the WAP only considered two arrangements (i.e., assignments of items to locations), which are given for the old and new arrangements. In contrast, the CSP has more than two arrangements. The number of arrangements is based on the number of times the layout changes during the outage. In addition, in the CSP, there are multiple temporary storage locations and more than one item can be assigned to those locations. In contrast, only one fixed temporary location is defined, which has a capacity of M items, in the WAP.

3.4 Other Related Problems

Beside the problems discussed above, there are other problems related to the CSP, or CSP can be applied to these problems. These problems try to find the routes for the material handling equipment with additional constraints.

3.4.1 Warehouse Storage/Retrieval Problem

An automated storage/retrieval system (AS/RS) is high-bay warehouse with storage/retrieval machines or automated stacker cranes that perform the storage and retrieval of storage modules (such as pallet or containers) (see **Van den Berg, 1999**). AS/RS are widely used in warehouses and distribution centers around the world. Some advantages of AS/RS over traditional warehousing systems are high space utilization, reduced labor costs and improved inventory control. An AS/RS consists of one or multiple parallel aisles with storage racks alongside. Usually, in every aisle a Storage/Retrieval (S/R) machine travels and performs the storage and retrieval of goods. With respect to product retrieval, unit load retrieval systems and order picking systems are distinguished. In a unit load retrieval system complete unit-loads are retrieved. Accordingly, the vehicles either perform one stop (storage or retrieval) or two stops (storage followed by retrieval) in a single trip. These trips are referred as a single command cycle and dual command cycle, respectively. In an order-picking system typically less then unit load quantities are picked, so that there will be multiple stops per trip (multiple-command cycle).

3.4.2 Crane Scheduling Problem

Lim *et al.* (2002) proposed the crane scheduling problem. It mainly considers how to schedule cranes in a port. It assumes that ships can be divided into holds and that cranes can move from hold to hold but that only one crane can work on one hold or job at any one time. This important component of port operations management is studied when certain spatial constraints, which are common to crane operations, are considered. The objective is to find a crane-to-job matching which will maximize throughput for such operations under these basic spatial constraints.

3.5 Conclusion

Based on the factors and related problems discussed in this chapter, the CSP is a more general and complex problem. The model and techniques for the CSP can be applied to other problems with little modifications. In other words, the CSP can be applied to some of the above TSPs by relaxing the location capacity constraint or preemption condition. Without these constraints, the crane will move the resource to its assigned location regardless of how many resources are in that location at that time. Then, each resource to be relocated is moved only once. However, space is considered an important scarce resource and temporary storage of resources is necessary in many industrial applications. Therefore, relaxing constraints of the CSP will result in infeasible routes in some applications

The main objective of the CSP is to find routes for a crane to relocate (pickup and deliver) resources in multiple periods in order to minimize the total crane travel cost and loading/unloading cost. Although the CSP is related to the TSP problems above, it has its own unique features. First, multiple periods are considered. There is one resource assignment at each time period. From the beginning of the second period, the routes in which the crane is to move the resources should be determined. The initial position of the crane is the last position crane visited in previous period. Second, the capacity of locations and temporary storage spaces are considered in the CSP. Any available location can be used as a temporary storage location for intermediate drops. Whenever a destination location is at its capacity or the crane route can be improved (i.e., travel cost can be decreased), a resource may be dropped off at a temporary storage space. Last, after sequences of resources to be moved are determined, a detail move plan, using a serial method, should be generated for the crane to indicate its actual movements.

CHAPTER 4

METHODOLOGY

4.1 Introduction

CSP is hard combinatorial optimization problem. Because of the location capacity, preemption, and multiple periods considered in the CSP, it is more general than some variants of the well-known TSP, such as the SCP. In this section, a mathematical model is formulated as an exact method to solve the CSP optimally. However, by using a mathematical model, only very small size CSP can be solved optimally in a reasonable amount of time. Therefore, heuristics, such as simulated annealing (SA), hybrid ant systems (HAS) heuristics, tabu search (TS), probability tabu search (PTS), tabu search with strategies (TS/S), as well as construction algorithms are proposed to solve the CSP in shorter amount of time with good solution qualities.

4.2 Exact Method

4.2.1 Mathematical Model and Notation

In this section a binary integer linear program is presented for the CSP. First the notation is given below. The indices are as follows.

$i, j, h = 1,..., L$: L is the number of locations;

$r = 1,..., R+1$: R is the number of items (or resources) and R + 1 represents a dummy item (empty move);

$k, k2 = 1,..., K$: K is an upper bound on the number of moves;

$t = 1,...,T$: T is the total number of periods;

The parameters are given below.

A_t = Set of items to be moved at period *t* (not including *R*+1);

B_t = Set of items to be moved at period *t* including *R*+1 ($A_t \cup \{R+1\}$);

Q_t = Set of locations which cannot be used as temporary storage locations because activities are being performed in those locations at period *t*;

UB_t = Upper bound for the number of moves at period *t*;

d_{ij} = Distance between locations *i* and *j*;

l_{rt} = The location of item *r* at period *t*;

n_{jt} = The number of items in location *j* at period *t*;

p = The initial location of the crane at the beginning of period 2 (assume location 1);

C_j = Capacity of location *j*;

M = A larger number;

W_1 = Travel cost per distance unit;

W_2 = Loading/unloading cost per item moved;

The decision variables are defined as

$$x_{t,k\in UB_t,r\in B_t,i,j} = \begin{cases} 1 \text{ If at the end of period } t, \text{ item } r \text{ is moved from location } i \text{ to } j \text{ at } k\text{th crane move;} \\ 0 \text{ Otherwise;} \end{cases}$$

The binary integer linear program for the CSP is as follows.

Minimize total cost =

$$W_1 * \sum_t \sum_{k\in UB_t} \sum_{r\in B_t} \sum_i \sum_j d_{ij} * x_{t,k,r,i,j} \quad + W_2 * \sum_t \sum_{k\in UB_t} \sum_{r\in A_t} \sum_i \sum_j x_{t,k,r,i,j} \qquad (0)$$

Subject to:

$$\sum_{k<=k2} \sum_i \sum_j x_{t,k,r,i,j} \le M * \sum_{k<=k2} \sum_j x_{t,k,r,i=l_{rt},j} \qquad \forall t, k2, r \in A_t \qquad (1)$$

$$\sum_{k}\sum_{j} x_{t,k,r,i=l_{rt},j} = 1 \qquad \forall t, r \in A_t \tag{2}$$

$$M*(1-\sum_{k<=k2}\sum_{i} x_{t,k,r,i,j=l_{rt+1}}) \ge \sum_{k>=k2+1}\sum_{i}\sum_{j} x_{t,k,r,i,j} \qquad \forall t, k2, r \in A_t \tag{3}$$

$$\sum_{k}\sum_{i} x_{t,k,r,i,j=l_{rt+1}} = 1 \qquad \forall t, r \in A_t \tag{4}$$

$$n_{jt} + \sum_{k<=k2}\sum_{r\in A_t}\sum_{i} x_{t,k,t,i,j} - \sum_{k<=k2-1}\sum_{r\in A_t}\sum_{i} x_{t,k,t,j,i} \le C_j \qquad \forall t, k2, j \tag{5}$$

$$\sum_{r\in B_t}\sum_{i}\sum_{j} x_{t,k,r,i,j} \le 1 \qquad \forall t, k \tag{6}$$

$$\sum_{r\in B_t}\sum_{j} x_{t=1,k=1,r,i=p,j} = 1 \tag{7}$$

$$\sum_{r\in B_t}\sum_{i} x_{t,k,r,i,j=h} \le \sum_{r\in B_t}\sum_{j} x_{t,k+1,r,i=h,j} + 1 - \sum_{r\in B_t}\sum_{i}\sum_{j} x_{t,k+1,r,i,j} \qquad \forall t, h, k < K \tag{8}$$

$$\sum_{r\in B_t}\sum_{i} x_{t-1,k=k2,r,i,j=h} - \sum_{r\in B_t}\sum_{i}\sum_{j} x_{t-1,k=k2+1,r,i,j} \le \sum_{r\in B_t}\sum_{j} x_{t,k=1,r,i=h,j} \qquad \forall t > 2, h, k2 \tag{9}$$

$$\sum_{i} x_{t,k=k2,r,i,j=h} \le \sum_{k>k2}\sum_{j} x_{t,k,r,i=h,j} \qquad \forall t, r \in A_t, h \ne l_{rt+1}, k2 \tag{10}$$

$$\sum_{i}\sum_{k<k2} x_{t,k,r,i,j=h} \ge \sum_{j} x_{t,k=k2,r,i=h,j} \qquad \forall t, r \in A_t, h \ne l_{rt}, k2 \tag{11}$$

$$\sum_{r}\sum_{i}\sum_{j} x_{t,k,r,i,j} \ge \sum_{r}\sum_{i}\sum_{j} x_{t,k=k+1,r,i,j} \qquad \forall t, k < K \tag{12}$$

$$\sum_{k}\sum_{r\in A_t}\sum_{i} x_{t,k,r,i,j=Q_t} = 0 \qquad \forall t, Q_t \tag{13}$$

$$x_{t,k,r,i,j} = 0 \text{ or } 1 \qquad \forall t, k, r, i, j \tag{14}$$

The objective function minimizes the sum of the total travel and loading/unloading costs, where the first term considers total travel cost and the second term considers total loading/unloading cost. Constraints (1) ensure that no item (or resource) can be moved before it is moved from its initial position, and constraints (2) ensure that each item being rearranged is always moved from its initial location. Similarly, constraints (3) ensure that no item can be moved after it is moved to its destination location, and constraints (4) guarantee that each item being rearranged is always moved to its destination location. Constraints (5) restrict an item from being moved to a location at full capacity.

Constraints (6) ensure that for each possible crane move, the crane can move only between two locations and can carry at most one item. Constraint (7) makes sure that the crane starts from its initial location *p* at the first period. Constraints (8) guarantee that each crane move destination is the starting position of the next crane move. For instance, if a crane move from location 1 to location 5, then the starting position of the next crane move is location 5. Constraints (9) ensure that the last crane move destination in period *t* – 1 is the initial position of the first crane move in period *t*. For example, if the last move in period 3 terminates at location 4, then the first move in period 4 starts at location 4. Constraints (10) and (11) are used to temporarily store items in temporary storage locations. Constraints (10) ensure that an item not at its destination location (i.e., either at its initial location or a temporary location) at the current move is moved to its destination location later. Constraints (11) allow an item to be moved to a temporary storage space. Constraints (12) ensure that once a crane starts to rearrange items in a period, it does not stop until all items have been rearranged. Constraints (13) restrict workspaces, currently used to perform activities, from being used as temporary storage spaces. Last, the restrictions on the decision variables are given in constraints (14).

It is important to not that this formulation does not allow items to be stored temporarily at their initial locations at any time period. These conditions should rarely happen if ever.

4.3 Heuristic Methods

As mentioned previously, large- or even medium-size CSPs cannot be solved optimally in reasonable computation time. Therefore, a heuristic approach needs to be used to obtain "good" solutions quickly. Hence, in this research, construction algorithms, hybrid ant system (HAS) heuristics, Tabu Search (TS), Probabilistic Tabu Search (PTS) and TS with strategies (TS/S) are presented for the CSP. As stated earlier, a simulated annealing (SA) heuristic for the CSP was presented in **McKendall et al. (2006)**. However, the authors did not consider loading/unloading cost; therefore, the SA heuristic is modified and used to solve the CSPs presented in this research. Also, it is used to compare against the other heuristics presented. Before discussing the heuristics, a solution representation is defined for the CSP.

4.3.1 Solution Representation

In this research, π_t is used to represent an ordered list of items (i.e., a sequence of items) to be moved by the crane at the beginning of each period t (where $t = 2, \ldots, T$). In another words, π_t is a permutation of items needed to be moved at the beginning of period t and is represented as follows:

$\pi_t = (\pi_{1t}, \pi_{2t}, \ldots, \pi_{n_t t})$, for $t = 2, \ldots, T$

where π_{it} is the ith item moved by the crane in period t, and $\pi_{n_t t}$ is the last item to be moved by the crane in period t. Hence, the entire solution can be represented as

$\pi = \{\pi_2, \ldots, \pi_T\} = \{(\pi_{12}, \pi_{22}, \ldots, \pi_{n_2 2}), (\pi_{13}, \pi_{23}, \ldots, \pi_{n_3 3}), \ldots, (\pi_{1T}, \pi_{2T}, \ldots, \pi_{n_T T})\}$.

Therefore, each sub-solution $\pi_t = (\pi_{1t}, \pi_{2t}, \ldots, \pi_{n_t t})$ is an ordered list of items to be moved by the crane. For instance, consider the CSP instance described in section 2.4, the feasible solution in **Table 2.3** can be represented as $\pi = \{\pi_2, \pi_3\} = \{(1, 4), (1, 2, 5, 3, 6)\}$

where the total cost was determined to be \$79 (i.e., $f(\pi)$ = \$79). Note in some cases, the permutation may represent different possible solutions with different cost unless a specific algorithm is used that leads to a unique solution.

4.3.2 Determining Crane Routes using π

Consider the CSP instance given in section 2.4 (see **Figure 1.1, 1.2** and **2.3**). Recall, when temporary storage locations are used, the solution (i.e., the permutation of items to be moved) may not indicate the actual movement of the crane (e.g., an item is moved more than once). For example, in the previous CSP instance the solution $\pi = \{\pi_2, \pi_3\} = \{(1, 4), (1, 2, 5, 3, 6)\}$ indicates the order in which the crane should move the items in both periods 2 and 3. However, in period 3, item 1 was moved twice (first before item 2 and after item 6) since item 1 was stored in a temporary storage location. As a result, the solution π does not show the movement of items from temporary storage locations to either other temporary storage locations or destination locations. Therefore, a serial heuristic is used to determine the specific movement of the items by the crane, considering temporary storage locations. However, if items are not moved to temporary storage locations (i.e., items move directly to their destination locations), the solution π would give the actual sequence of item moves by the crane. Next, a serial heuristic is given to obtain the actual sequence of the crane so that the total cost of the solution can be obtained.

A serial heuristic is presented below for the CSP. As mentioned earlier, this heuristic is used to determine the actual sequence of items moved by the crane for each period.

Also, it is used to obtain the total cost of the solution. This heuristic is similar to a method that was used to schedule project activities with limited resources such that makespan is minimized (e.g., **Kolisch (1996)**). The steps for the serial heuristic are given as follows.

```
Step 1: Set $t = 1$, where $t$ is a period index;
        Initialize Objective Function Value (OFV) $f(\pi) = 0$;

Step 2: Set $t = t + 1$;
        Initialize set $RIT$ as empty, where $RIT$ is the set of resources (items) in
            temporarily storage space in period $t$;
        Obtain set $NRL$, where $NRL$ is the set of numbers indicating the number of
            resources (items) in each location at period $t - 1$;
        If $t = 2$, set crane current location $x = p$ (e.g. location 1);
        Else, set current location $x = \eta$, which is the last position of the crane in period $t$
            $- 1$;
        Set $i = 0$, where $i$ is the position index in $\pi_{it}$;

Step 3: Set $i = i + 1$;
        Set $r = \pi_{it}$, where $r$ is the resource in position $i$ in period $t$;
        Update $f(\pi) = f(\pi) + w_1 * d(x, S(r))$, where $S(r)$ is the location of item $r$ in period $t$
            $- 1$, $d(x, S(r))$ is the distance from location $x$ to $S(r)$, and $w_1$ is the cost per
            distance unit;
        Set $x = S(r)$;

Step 4: Obtain $N(r)$, the destination location of item $r$ in period $t$;
    a)  If $NRL(N(r)) = Cap$, where $Cap$ is the capacity of the locations,
          Crane will move item $r$ to the closest available temporary storage location $tl$
            (break tie by selecting an item with least item number), and insert item $r$ into
            set $RIT$;
        Update the $NRL$ such that
                $NRL(S(r)) = NRL(S(r)) - 1;$
                $NRL(tl) = NRL(tl) + 1;$
        Update $f(\pi) = f(\pi) + w_1 * d(x, tl) + w_2$, where $w_2$ is unloading/loading cost/item;
        Update $S(r) = tl$, and set $x = tl$;
    b)  Else, move item $r$ to its assigned (destination) location $N(r)$.
        Update set $NRL$ such that
                $NRL(S(r)) = NRL(S(r)) - 1;$
                $NRL(N(r)) = NRL(N(r)) + 1;$
        Update $f(\pi) = f(\pi) + w_1 * d(x, N(r)) + w_2$, and set $x = N(r)$;

Step 5: If $\exists\ r' \in RIT$, $x = S(r')$, and $NRL(N(r')) <$ Cap,
          Set $r = r'$, remove item $r$ from $RIT$, and go to Step 4b;

Step 6: If $i = n_t$, where $\pi_{n_t t}$ is the last item in the sequence $\pi_t$, go to Step 7;
        Else, go to Step 3;

Step 7: If $RIT \neq \emptyset$,
          Find item $r'' \in RIT$ for which $d(x, S(r''))$ is minimum (break tie by
              selecting an item with least item number) and its destination location is
              not at capacity.
          If the destination locations of the items in $RIT$ are at full capacity, select item
              $r'' \in RIT$ for which $d(x, S(r''))$ is minimum (break tie by selecting an
              item with least item number)
          Set $r = r''$;
          Update $f(\pi) = f(\pi) + w_1 * d(x, S(r))$, set $x = S(r)$, and remove item $r$ from $RIT$;
          Go to Step 4;
        Else, if $t < T$, go to Step 2;
          Else, terminate heuristic;
```

4.3.3 Construction Algorithms

In order to obtain diverse solutions for the CSP, three construction algorithms are proposed in this research. The first is a very simple algorithm which lists the items to be moved in ascending order for each period. For example, if items 1, 2, 4, and 8 need to be reassigned to new locations in period t, then $\pi_t = \{1, 2, 4, 8\}$. This construction algorithm is called CAI.

The second construction algorithm, called CAII, is a nearest neighbor heuristic. In other words, the order in which the items are moved is based on the distances between the current location of the crane and the locations of the items to be moved. For instance, if items 1, 2, and 4 are reassigned to locations in period t and the distances between the current location of the crane and the locations of the items are 3, 2, and 1, respectively, then item 4 is assigned to the first position of the move sequence. If a tie exists, the item with the least number of items in its destination location is selected. Next, the item assigned to the second position of the sequence is the item closest to either the destination location or the temporary storage location of item 4. Nevertheless, the item closest to the current location of the crane is selected, say for instance item 1. As a result, $\pi_t = \{4, 1, 2\}$. It is obvious that this heuristic attempts to minimize crane travel cost.

In the third construction algorithm, CAIII, the location of the first item to be moved is selected where the location of the item has the most items to be moved. If a tie exists between one or more locations, the location closest to the crane is selected. Once this location is determined, the item in this location with the least number of items in its destination location is selected first. This process is repeated for all items needed to be moved. This heuristic attempts to minimize the use of temporary storage locations such that loading/unloading costs are minimized.

4.3.4 Local Neighborhood Search Techniques

As stated previously, the CSP is to find the sequences in which a single overhead crane moves items to their assigned locations with respect to minimizing the total crane travel cost and loading/unloading cost. Hence, if n_t is the total number of items to be moved at each period t and T is the total number of periods in the CSP, the solution space consists of $\prod_{t=2}^{T}(n_t!)$ possible sequences. In order to improve a current solution π, a method to modify the sequences (i.e., find a neighboring solution) needs to be developed. In this research, neighboring solution is obtained by exchanging the positions of two items to be moved in π_t. Next, local neighborhood search techniques are used to improve the solution. Generally, a local neighborhood search technique starts with an initial solution π and keeps improving this solution until no further improvement can be found. Two local search techniques, random descent method and steepest descent method, and their neighborhood structure are presented in this section.

a) Random Descent Method

This local neighborhood search technique starts with an initial solution π, as the current solution, and randomly selects a neighboring solution π' in the neighborhood of

π, $\pi' \in N(\pi)$. A neighboring solution π' is obtained by randomly selecting a period t ($1 < t \leq T$) and randomly exchanging the locations of two items in π_t. If $f(\pi') \leq f(\pi)$, then π' becomes the current solution (set $\pi = \pi'$). Otherwise, the current solution π does not change. This process is repeated until a predefined stopping criterion has been met. This technique is known as the random descent heuristic. Since only improving solutions are accepted during the search process, this local search technique as well as others (e.g., steepest descent, first improvement), converge to the local optimum of the initial solution. Often times, this may result in a low-quality solution. Hence, this method does not guarantee a global optimum, but it is used within many meta-heuristics such as the proposed SA heuristic and HAS heuristics which are used to improve the performances of local neighborhood search techniques. More specifically, the random descent heuristic is used to search the neighborhoods of solutions, and other components of the proposed SA or HAS heuristics will be used to accept non-improving solutions so that the global optimum may be obtained.

b) Steepest Descent Method

The steepest descent heuristic starts from an initial solution π and explores its entire neighborhood, $N(\pi)$. In other words, all possible pairwise exchanges between items to be moved are considered for each period t, and the best exchange is performed. That is, all the neighboring solutions in the neighborhood of π, $N(\pi)$, is considered for each period t and the best neighbor $\pi' \in N(\pi)$ (i.e., the best move, *move** = (t, u, v), which exchange the locations of items u and v in period t) is selected such that $f(\pi') \leq f(\bar{\pi}) \,\forall\, \bar{\pi} \in N(\pi)$.

The corresponding solution is the current solution at the next iteration (i.e., $\pi = \pi'$). When a local optimum is obtained (i.e., no π' exist such that $f(\pi') \leq f(\bar{\pi})$ for $\forall\ \bar{\pi} \in N(\pi)$), the heuristic terminates. Therefore, the steepest descent heuristic accepts only improved solutions and do not accept non-improving (uphill) solutions as with other simple local search techniques such as the random descent heuristic. As a result, the steepest descent often converges to a poor local optimum, usually depending on the quality of the initial solution π. Therefore, different techniques (e.g., short term memory, aspiration criterion) are used in the heuristics (e.g. tabu search) to overcome these drawbacks of the simple steepest descent local search technique in search of the global optimum. Next, the proposed heuristics are presented for the CSP.

4.3.5 Simulated Annealing Heuristic

SA heuristic is a meta-heuristic used to solve many combinatorial optimization problems. **Kirkpatrick (1983)** was the first to use SA to solve combinatorial optimization problems. The major advantage when comparing SA with local search method described above is that SA allows for the escape from local optimum (i.e., allowing accepting non-improving solutions), with the possibility of reaching a global optimum, by allowing uphill moves. Its ability of escaping from local optimal solutions is based on analogy with a method of cooling metal which is known as “annealing”.

SA presented for the CSP starts with an initial solution π by using a construction algorithm. A neighboring solution π' is then generated by randomly exchanging the positions of two items at period t. If $f(\pi') < f(\pi)$, then π' is accepted as the current solution, else it is accepted with probability $\exp(-[f(\pi') - f(\pi)]/Temp)$, where *Temp* is a temperature parameter that is typically non-increasing at each iteration, *iter*. Initially, the probability of allowing a non-improved move by probability $\exp(-[f(\pi') - f(\pi)]/T_0)$ will be relatively large. However, as *Temp* decreases, this probability also decreases. Thus, as a high quality solution is obtained, non-improved moves are less likely to be accepted. **McKendall and Shang (2006)** applied a SA heuristic to the CSP, which considered the objective of minimizing the crane travel cost. However, as mentioned earlier, in this research, the objective function is modified to consider both crane travel cost and loading/unloading cost. The steps for SA heuristic are as follows.

Step 1: Set parameters and counters:

T_0 = initial temperature;

ε = cooling ratio;

H_0 = number of iterations performed at initial temperature epoch length;

γ = parameter used to increase epoch length;

T_f = final temperature;

π = current solution;

π^* = best solution found;

Set current temperature $Temp = T_0$;

Set current epoch length $H = H_0$;

Set iteration counter at each temperature $ic = 0$;

Set $\pi^* = \pi$;

Step 2: Set $ic = ic + 1$;

Generate a neighboring solution π' in the neighborhood of current solution π by the random pairwise exchange (i.e., $\pi' \in N(\pi)$).

```
Step 3:  If $f(\pi') \le f(\pi^*)$, then set $\pi^* = \pi'$, $\pi = \pi'$, and go to step 4;
         Else,
             if $f(\pi') \le f(\pi)$, then set $\pi = \pi'$;
             Else, the probability of accepting $\pi'$ is $\exp(-[f(\pi') - f(\pi)]/Temp)$;

Step 4:  If $ic < H$, then go to step 2;
         Else, set $Temp = \varepsilon Temp$ and $H = H(1 + \gamma)$;
             If $Temp > T_f$, set $ic = 0$ and go to step 2;
             Else, terminate heuristic;
```

First, the temperature *Temp* has been set to the initial value T_0 at the beginning. The initial temperature T_0 was determined by using the formula $\exp(-[f(\pi') - f(\pi)]/T_0) =$ 0.25 and $f(\pi') - f(\pi) = 0.1 * f(\pi)$, where 0.10 and 0.25 were obtained experimentally. In other words, $T_0 = -0.10 * f(\pi)/\ln(0.25)$ and the initial probability of accepting a non-improving solution, which is 10% worse then solution π, is 25%. Then, *Temp* is decreased by *εTemp* (i.e., *Temp* = *εTemp*) at every temperature reduction. In other words, after performing a number of random pairwise exchanges, say *H*, the temperature is reduced by *εTemp*. The epoch length *H* is the number of iterations (random pairwise exchanges) performed at each temperature. At the initial temperature T_0, the number of random pairwise exchanges is the number of periods (i.e., $H_0 = T$). Otherwise, the number of random pairwise exchanges is $H(1 + \gamma)$, as in the SA heuristic presented by **Bouleimen and Lecocq (2003)** for the resource constrained project scheduling problem. The process is continued until a predefined stopping temperature has been met (i.e., $Temp \le T_f$). Initially, the probability of allowing a non-improved move,

$\exp(-[f(\pi') - f(\pi)]/Temp)$, will be relatively large. However, as *Temp* decreases, this probability also decreases. The values of γ and ε are obtained experimentally and explained in the computational results section.

4.3.6 Hybrid Ant System Algorithms

Gambardella *et al.* (1999) presented an ant system optimization, called HAS-QAP, used to improve randomly generated solutions. HAS is a modification of the ACO heuristics presented in **Colorni *et al.* (1991)** and **Dorigo and Gambardella (1997)**. In both papers, the ACO heuristics were used to construct solutions for the TSP. **Dorigo *et al.* (1999)** discuss the applications of ant systems to construct solutions for some well-known problems such as the TSP and vehicle routing problem. Several authors presented ant systems for the quadratic assignment problem (QAP) such as **Maniezzo and Colorni (1999)**, **Stutzle and Dorigo (1999)**, and **Gambardella *et al.* (1999)**. The ant colony optimization (ACO) and hybrid ant system (HAS) heuristics use the idea of how ants search for food and leave a chemical substance called pheromone so that other ants can find the food source, to solve combinatorial optimization problems. The pheromone trails in these ant systems serve as distributed, numerical information which the ants use to probabilistically construct (as in ACO) or improve (as in HAS) solutions to combinatorial optimization problems.

Artificial ants used in HASs are stochastic solution improvement procedures that probabilistically improve solutions. In other words, the artificial ants are equipped with a local search heuristic function to guide their search through the set of feasible solutions.

HAS also provides mechanisms to either intensify or diversify the search. **McKendall and Shang (2006)** modified the HAS-QAP heuristic and presented three HASs for the dynamic facility layout problem (multi-period QAP). The first heuristic is a direct modification of HAS-QAP for the dynamic facility layout problem. The second heuristic uses a simulated annealing heuristic, instead of a random descent heuristic as in HAS-QAP, to improve the solutions obtained after performing pheromone trail swaps. The third heuristic is exactly like the first, except that the random descent heuristic has a look-ahead/look-back strategy. In this research, two HASs are presented for the CSP. The first is similar to the HAS-QAP (use random descent heuristic as a local search technique), and the second uses simulated annealing instead of a random descent heuristic. The first HAS, called HAS I, is briefly summarized below.

Step 1: Randomly generate a set of initial CSP solutions, called Ω, where $\pi \in \Omega$. That is, list items to be moved in random order in each period for each ant, π^m, where m =1, 2, …, M.

Step 2: Set paremeters and counters:

M = number of ants;

Q = parameter used to determine the initial values of the pheromone trail matrix P;

E = number of exchanges used to modify each CSP solution;

q = the probability that pheromone trail exchange policy 1 will be selected to modify a CSP solution. As a result, policy 2 is chosen with a probability of $1 - q$; The policies will be discussed later.

α_1, α_2 = parameters used to weaken and enforce pheromone trails, respectively;

S = number of consecutive iterations, without improving best solution π^*, before implementing diversification strategy;

Set *intense* = 1 (intensification strategy is active);

run_time = run time of heuristic;

Step 3: Use the random descend method presented above to improve all M solutions and obtain improved set of solutions Ω'. Set $f(\pi^*) = \min\{f(\pi) \mid \pi \in \Omega'\}$.

Step 4: Initialize pheromone trail matrix P:

Set $p_{rjt} = Q/f(\pi^*)$ $\forall r \in A_t, j, t$, where the entries p_{rjt} measures the desirability of assigning item $r = \pi_{it}$ (item r in ith position in period t) to be moved to the jth position in π_t of π and A_t is a set of items to be moved.

Step 5: Modify each solution in set Ω' by using the pheromone trail matrix and one of two policies to obtain another set of solutions $\overline{\Omega}$. More specifically, for each ant $\pi \in \Omega'$:

```
    set counter $e = 1$;
    Step 5 a):
        If $e < E$,
            Randomly select a period $t$ and an item $x \in \pi_t$, say $\pi_{ut}$;
            Randomly generate a number $w$ between 0 and 1;
            If $w \le q$,
                then use policy 1. That is, select new position $v \neq u$ for item $x$
                such that $p_{xvt} + p_{yut}$ is maximized where item $y$ is currently in
                position $v$ in period $t$ (i.e., $y = \pi_{vt}$);
            Else,
                use policy 2. That is, $v \neq u$ is chosen with a probability of
                $\frac{p_{xvt} + p_{yut}}{\sum_{v \neq u}(p_{xvt} + p_{yut})}$ where $y$ is currently in position $v$ in period $t$
                (i.e., $y = \pi_{vt}$);
```

Exchange the locations of items x and y in π_t;

Set $e = e + 1$, and go to the beginning of step 5 a);

Else, all solutions $\pi \in \Omega'$ have been modified such that the new set of solutions is $\overline{\Omega}$; therefore, go to step 6.

Step 6: Use the random descent heuristic to improve solutions in the set $\overline{\Omega}$ and obtain the set of solutions Ω''.

If there exist $\pi \in \Omega''$ such that $f(\pi) < f(\pi^*)$, then

set $\pi^* = \pi$ where $f(\pi^*) = \min\{f(\pi) \mid \pi \in \Omega''\}$;

Step 7: If $itw = S$, set $itw = 0$, and randomly generate a set of $M - 1$ solutions. The $M - 1$ solutions and π^* make up a diverse set of solutions Ω, and go to Step 3;

Step 8: If there exists an π^m such that $f(\pi^m \in \Omega'') < f(\pi^m \in \Omega')$, then set $temp = 1$;

Else, set $temp = 0$;

If $intense = 1$,

For each ant π^m, update new solution based on intensification strategy. That is, update Ω' (for next iteration) where each solution $\pi^m \in \Omega'$ is such that $\min\{f(\pi^m) | \pi^m \in \Omega' \cup \Omega''\}$.

Else, set $\Omega' = \Omega''$.

Set $intense = temp$;

Step 9: Update pheromone trail matrix as discussed below.

Step 10: If heuristic run time $\geq$ *run_time*, then terminate the heuristic.

Else, go to step 5;

In step 3, the randomly generated solutions obtained in step 1 are improved using the random descent heuristic. Based on the value of the best solution obtained in step 3, the pheromone trail matrix is initialized in step 4. In step 5, a period t and an item x in position u is randomly selected. The order in which the crane will move this item is exchanged with another item y in position v which is selected based on one of two policies. The first policy uses the pheromone trail matrix to select the item y in location v based on the most desirable exchange between the item x in position u and all other items. The second policy selects an item y in location v randomly, but there is a higher probability that item y with the largest $p_{xvt} + p_{yut}$ will be accepted. After E exchanges using the pheromone trail matrix, the solutions are improved using the random descent heuristic in step 6. If S iterations are performed without improving the best found solution, the diversification strategy is implemented in step 7. This strategy is used to explore diverse areas of the solution space. Otherwise, the intensification strategy is invoke in step 8, if it is active (i.e., *intense* = 1). If *intense* = 0, then the intensification strategy is not activated. Intensification is used to explore promising areas of the solution space. Also, the set of solutions Ω' for the next iteration is obtained. In step 9, the pheromone trail matrix P is updated such that only the best solution found so far is permitted to deposit pheromone. The pheromone trail matrix is updated according to **Dorigo and Gambardella (1997)**. Set

$$p_{rjt} = (1-\alpha_1)p_{rjt} + \alpha_2\delta(\pi_{rjt}) \quad \forall r \in A_t, j, t$$

where

$$\delta(\pi_{rjt}) = \begin{cases} 1/f(\pi^*) & \text{if } \pi_{rjt} \in \text{best solution } \pi^* \\ 0 & \text{otherwise} \end{cases}$$

The first terms in the expression, $(1-\alpha_1)p_{rjt}$, is used to weaken the pheromone trails where $0 < \alpha_1 < 1$ is a parameter used to control the evaporation rate of the trails. A value close to zero indicates the trails will remain active for a longer time, and a value close to one indicates a high degree of evaporation and a shorter memory of the system. However, the expression $\alpha_2\delta(\pi_{rjt})$ is used to strengthen the pheromone trails where α_2 is a parameter used to control the reinforcement of the pheromone trails corresponding to the best solution π*. Using only the best solution π* speeds up the convergence of the heuristic (**Gambardella *et al.,* 1999**). In step 10, the heuristic is terminated, if heuristic run time is at least *run_time* minutes.

The second HAS, called HAS II, uses the simulated annealing heuristic presented earlier, instead of the random descent heuristic, to improve solutions initially and after performing *E* pheromone trail exchanges. **McKendall and Shang (2006)** presented a similar HAS for the dynamic facility layout problem, which out-performed the HAS with a random descent heuristic. Therefore, it is applied to the CSP presented in this research. More specifically, the random descent heuristic presented in steps 3 and 6 are replaced by the SA heuristic presented previously in section 4.3.5.

4.3.7 Tabu Search Heuristics

The Tabu Search (TS) heuristic was first proposed by **Glover (1986)** and presented in detail in **Glover (1990a) and (1990b)** and **Glover and Laguna (1992)**. The basic idea of TS is to improve a solution iteratively, using some guiding rules such as dynamic tabu list size as well as intensification and diversification strategies to obtain good solutions in complex solution spaces. The basic components of the TS heuristic are discussed below.

The TS heuristic uses the steepest descent heuristic with short term memory (or recency based memory) to accept uphill moves. In other words, the steepest descent heuristic converges to a local optimum; however, short term memory is used to forbid the recent moves so that the heuristic can climb out of the valley which contains the local optimum (i.e., accept uphill moves) in search of better local optima. In the CSP, if the best solution in the neighborhood of the current solution π is π' (i.e., $f(\pi') \le f(\bar{\pi})$ for $\forall\ \bar{\pi} \in N(\pi)$) and π' is obtained by $move^* = (t, u, v)$, which exchanges the locations of items u and v in period t, then this move is **tabu restricted** for a certain duration (*tabusize*), called **tabu list size**. For the CSP, the tabu list size $tabusize_t$ are different for different periods. The tabu status and tabu list size of each move are maintained in the lower half of the tabu list structure $tabu[t][u][v]$, where $u > v$. Sometimes a move which is tabu restricted may give the best solution found thus far. Therefore, the **aspiration criterion** is used to override the tabu restriction of a move when the move improves the best found solution thus far. For example, if at the current iteration (*iter*), items 2 and 6 exchange positions in period 3 (i.e., $move^* = (3, 6, 2)$) where $\pi = \{\pi_2, \pi_3\} = \{(1, 4), (1, 2, 5, 3, 6)\}$, then $\pi' = \{\pi_2, \pi_3'\} = \{(1, 4), (1, 6, 5, 3, 2)\}$ and $tabu[3][6][2] = tabusize_3 + iter$. Therefore, the move, which considers exchanging items 2 and 6 in period 3, is tabu until $iter = tabusize_3 + iter$. In other words, the move, which considers exchanging items 2 and 6 in period 3, can be performed again when $iter = tabusize_3 + iter + 1$. Also, if *move** had been performed recently, is tabu restricted, and it improves the best solution found thus far, then the aspiration criterion is used to override its tabu restriction. Any move which is acceptable (e.g., nontabu move and tabu move overridden by aspiration criterion) is defined as an **admissible move**. Hence, *move** is defined as the best admissible move. A simple TS heuristic is outlined below.

```
Step 1: Initialize parameters and counters:
        $T$ is the total number of periods;
        $Tabu[][][]$ is the tabu list structure;
        $tabusize_t$ is the tabu tenure length for period $t$;
        $iter$ is iteration number where $iter = 0$;
        $TRT$ is total running time before terminating the heuristic;

Step 2: Obtain an initial solution $\pi$ by using a construction algorithm and determine its
        objective function value $f(\pi)$;

Step 3: Set $\pi^* = \pi$, where $\pi^*$ is the best solution found thus far and set $f(\pi^*) = f(\pi)$;

Step 4: Set $iter = iter + 1$;
        Find the best admissible move $move^* = (t, u, v)$ with respect to objective
        function value, which gives $\pi'$;

Step 5: Set $\pi = \pi'$ and $f(\pi) = f(\pi')$;
        If $f(\pi) < f(\pi^*)$,
            Set $\pi^* = \pi$ and $f(\pi^*) = f(\pi)$;

Step 6: Update tabu list as $tabu[t][u][v] = iter + tabusize_t$;

Step 7: Stopping Criteria
        If heuristic running time < $TRT$, go to Step 4.
        Else, terminate the heuristic and return solution $\pi^*$ and total cost $f(\pi^*)$;
```

4.3.8 Probabilistic Tabu Search

The probabilistic tabu search (PTS) heuristic for the CSP is an extension of the TS heuristic presented in the previous section. Although the TS heuristic accepts uphill moves to escape from poor local optima, it is a deterministic heuristic which may not explore a large portion of the solution space far away from the initial solution. Therefore, the PTS heuristic is used to add randomness so that diverse solutions may be obtained. The main difference between the PTS and TS heuristic is how *move** is selected at each iteration. Before, in the TS heuristic, *move** is defined as the best admissible move. However, in the PTS heuristic, *move** is selected randomly among the best G admissible moves. Similarly, *move** is used to generate the new solution π', which becomes the current solution π for the next iteration. More specifically, for the PTS heuristic, *move** is selected from the best G admissible moves according to their probabilities. In other words, the best G admissible moves are sorted based on their corresponding OFV. The probability to accept the first (i.e., best) move from the G moves is p. In this research, G is set to 10 and p is set to 0.4. If the first move is rejected, the second move is accepted with a probability $p(1 - p)$. This process is repeated until a move is selected. However, if no move is selected after considering all G moves, the first move will be selected. An improved technique for selecting *move** is to use the cumulative probability proposed in **Chiang and Chiang (1998)**. More specifically, the cumulative probability table below is created using the following equations.

$$CP(i) = \begin{cases} 0, & \text{for } i > G \text{ or } i < 1, \\ p(1-p)^{G-1}, & \text{for } i = G, \\ AP(i+1) + p(1-p)^{i-1}, & \text{for } 2 \le i < G, \\ 1, & i = 1 \end{cases}$$

i	***P***	***AP***
0	0	0
1	0.4	1
2	0.24	0.5939
3	0.144	0.3539
4	0.086	0.2099
5	0.0518	0.1235
6	0.0311	0.0717
7	0.0186	0.0406
8	0.0111	0.0219
9	0.0067	0.0107
10	0.0040	0.0040
>10		0

Table 4.1: Cumulative probabilities with G =10 and p = 0.4.

Then, a random number x between 0 and 1 is generated. If $CP(i + 1) < x \leq CP(i)$, then the ith move out of the G moves is selected as *move**. In the cumulative probability table presented above, where G = 10 and p = 0.4. For instance, a random number x, say x = 0.25 is generated. Since $CP(4)$ =0.2099 < $x \leq CP(3)$ = 0.3539, the 3rd move from the list of G moves is defined as *move**. Therefore, the PTS heuristic contains steps 1 – 7 of the TS heuristic. However, *move** in step 4 is obtained using the cumulative probability table and the equations defined above.

Although the PTS heuristic usually out-performs the basic TS heuristic, only a few researchers actually applied the PTS heuristic to combinatorial optimization problems in the literature. **Chiang and Chiang (1998)** applied the PTS heuristic to a quadratic assignment problem. **Lim *et al* (2004)** presented a similar PTS heuristic to solve a crane scheduling problem with spatial constraints.

4.3.9 Tabu Search with Strategies

The main idea of the TS heuristics discussed above, TS and PTS heuristics, is as follows. The use of short term (recency-based) memory to keep track of the most recent moves is to avoid getting trapped at a local optimum by allowing uphill (non-improving) moves. More specifically, the steepest descent heuristic with recency-based memory and the aspiration criterion defined in this research is used to obtain the best admissible move *move** in the TS heuristic. In the PTS heuristic, the same components are used to obtain G admissible moves, as in the TS heuristic. However, the admissible move selected, *move**, may not be the best move but is in the top G admissible moves. More specifically, *move** is selected based on cumulative probabilities. In retrospect, the proposed TS heuristic is a deterministic heuristic in which solution quality may depend on diverse initial solutions. However, the PTS heuristic adds randomness to the TS heuristic such that the solution quality does not depend on the initial solutions provided.

Adding cumulative probabilities to the basic TS heuristic is a diversification strategy used to better explore different areas of the solution space. Other diversification techniques such as dynamic tabu list size and frequency-based (long-term) memory may be used to improve the performance of the basic TS heuristic. Also, intensification strategies may also be used to explore promising regions within the solution space more thoroughly. Next, the diversification and intensification strategies used to improve the basic TS heuristic for the CSP are discussed below.

a) Diversification Phase

The ideas of diversification in **Chiang and Kouvelis (1996)** are used in this research. There are the dynamic tabu list size and frequency-based memory. With dynamic tabu list size, the tabu list size *$tabusize_t$* is not fixed as in the basic TS procedure but is dynamic in order to diversify the search. In other words, tabu list size *$tabusize_t$* is dynamic and updated at each iteration. More specifically, the *$tabusize_t$* varies between a lower bound LB_t and an upper bound UB_t. To illustrate the dynamic tabu list size, first, an improvement of the total cost of the current solution is calculated as:

$$z = 1 - \frac{f(\pi')}{f(\pi)}$$

where π is the current solution and π' is a neighboring solution obtained by performing the best admissible move *move** in period t. Then, LB_t and UB_t are defined as:

$$LB_t = n_t * T / 3 \quad \forall t > 1$$

$$UB_t = n_t * T * 2 \quad \forall t > 1$$

where n_t is the total number of items to be moved at each period t and T is the total number of periods in the CSP. Since n_t is changed between periods, LB_t, UB_t and *$tabusize_t$* in different periods may not be same. Initially, the tabu list size *$tabusize_t$* in period t equals to LB_t. Then, at the end of each iteration, tabu list size *$tabusize_t$* will be updated as follows.

$$f(tabusize_t) = \begin{cases} tabusize_t & \text{if } z \le 0 \\ LB_t + \dfrac{UB_t - LB_t}{\alpha} * z & \text{if } 0 < z \le \alpha \\ UB_t & \text{if } \alpha < z < \beta \\ 3 * UB_t & \text{if } z \ge \beta \end{cases}$$

where α and β are pre-defined parameters. The graph of this function for $z > 0$ is shown in

Figure 4.1.

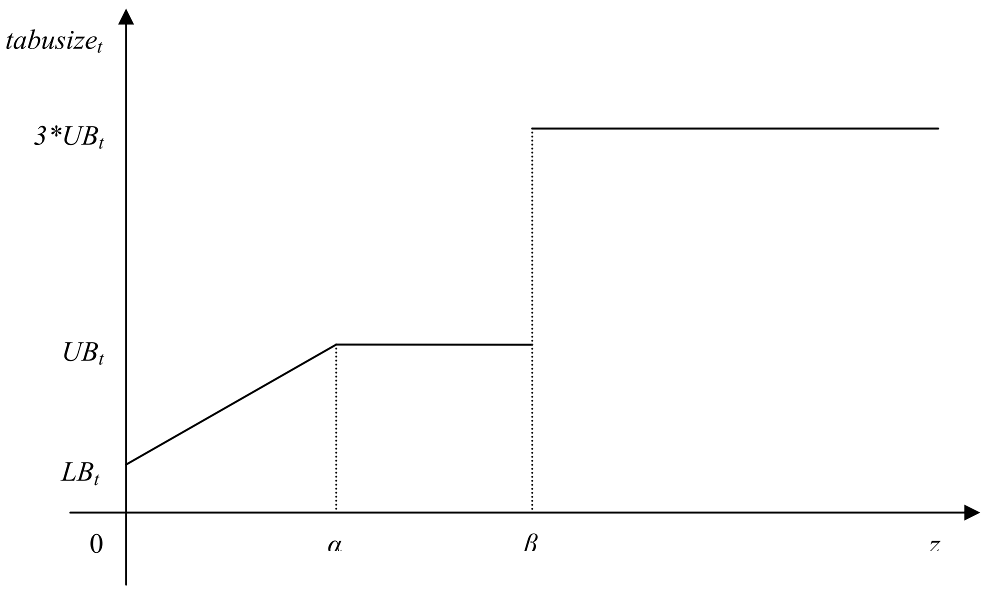

Figure 4.1: Dynamic tabu list size $tabusize_t$ when $z > 0$

Therefore, the basic TS heuristic can be easily modified to consider dynamic tabu list sizes by replacing step 6 with the following.

Step 6: Update $tabusize_t = f(tabusize_t)$;
Update tabu list as $tabu[t][u][v] = iter + tabusize_t$;

Another diversification technique is to use frequency-based (long term) memory. This diversification strategy is added to the TS heuristic to diversify the search space, forcing the search into unexplored regions of the solution space. It is employed only when no improving admissible move exists and a penalty is given for each non-improving move. In order to apply diversification strategy to the TS heuristic, a long term memory is needed. Unlike the short term (recency) memory discussed above, where only the recent moves are given, the long term memory keeps track of the frequencies of all moves during the search process and shows the distribution of each move. For the CSP, the long term memory is used to keep track of the number of exchanges between pairs of items. The frequencies of the moves are kept in the upper half of the tabu list structure *tabu*[*t*][*u*][*v*], where $u < v$. For instance, if items u and v exchanged positions in period t (i.e. *move** = (t, u, v)), then long term memory is updated as:

tabu[*t*][*u*][*v*] = *tabu*[*t*][*u*][*v*] + 1 for $u < v$;

See a tabu list structure instance below. The tabu structure in **Figure 4.2** shows that in period 3, 5 items are to be relocated (i.e., item 1, 2, 3, 5, and 6) and after several iterations, a number of exchanges have been performed. For instance, *tabu*[*3*][*3*][*5*] = 13, which means that items 3 and 5 exchanged locations 13 times thus far during the execution of the heuristic.

Tabu $t = 3$	Item				
Item	1	2	3	5	6
1	0	5	2	9	0
2	140	0	0	0	0
3	47	0	0	13	0
5	212	102	320	0	17
6	99	0	0	401	0

Figure 4.2: A tabu structure instance.

The long term memory structure, which is defined above, shows the distribution of moves in each period and is used in the diversification strategy to penalize non-improving moves by giving a larger penalty to the moves with greater frequency counts. Notice, in step 4 presented above, the basic TS heuristic selects the best admissible move *move** = (t, u, v) and the corresponding solution π' is obtained by considering $f(\pi') \leq f(\overline{\pi})$

$\forall\ \bar{\pi} \in N(\pi)$). However, if the penalty for non-improving moves is added, then a modified objective function is used. Thus, step 4 is modified as follows.

Step 4: Set *iter* = *iter* + 1;

Find the best admissible move *move** = (*t, u, v*) which gives π' according to a revised OFV where the revised OFV is

$rf(\pi') = f(\pi') + \lambda * tabu(t, u, v)$ (assume $u < v$)

$$\lambda = \begin{cases} 0 \text{ if } f(\pi') < f(\pi); \\ \Lambda \text{ Otherwise;} \end{cases}$$

Λ is a pre-defined parameter for penalty. For each non-improving move, a penalty is given based on the frequency of the move performed thus far. Then, *move** = (*t*, *u*, *v*) will be selected based on this revised evaluation value for each move.

b) Intensification Phase

Unlike the diversification strategies presented above, intensification explores promising areas of the solution space more thoroughly. In the literature, intensification typically operates by restarting from relatively high quality solutions or modifying a solution to favor some attributes. In this research, the intensification method described in **Chiang and Kouvelis (1996)** is applied and integrated into the proposed TS heuristic for the CSP. This intensification strategy is implemented by fixing two items after exchanging their locations, if this exchange reduces the total cost of the current best solution π^* by at least α, and α is the same value as the parameter used in the diversification strategy as used in **Chiang and Kouvelis (1996)**. Also, experimental tests were performed which gave the same results. Similar to the TS aspiration criterion, a fixed item can be freed to exchange its location with other items if the exchange results in an improvement better than the best solution found so far. Intensification is employed after a certain number (η) of iterations have been performed, since there are relatively higher probabilities of having more solution improvements at earlier iterations. The steps of the basic TS heuristic are modified for the diversification and intensification strategies discussed above and are as follows.

Step 1: Initialize parameters and counters:

T is the total number of periods;

Tabu[][][] is the tabu list structure;

$tabusize_t$ is the tabu tenure length for period t;

iter is iteration number where *iter* = 0;

TRT is total running time before terminating the heuristic;

LB_t is the lower bound for $tabusize_t$;

UB_t is the upper bound for $tabusize_t$;

α is the parameter for diversification and intensification strategy;

β is the parameter for diversification strategy;

Λ is a parameter for penalty

η is the number of iterations performed before invoking the intensification strategy;

F[][] stores all items fixed during the intensification process;

Step 2: Obtain an initial solution π by using a construction algorithm and determines its objective function value $f(\pi)$;

Step 3: Set $\pi^* = \pi$, where π^* is the best solution found thus far and set $f(\pi^*) = f(\pi)$;

Step 4: Set *iter* = *iter* + 1;

```
        Find the best admissible move move* = (t, u, v) which gives $\pi'$ with respect to
        a revised OFV where the revised OFV is
        $rf(\bar{\pi}) = f(\bar{\pi}) + \lambda * tabu(t, u, v)$ (assume $u < v$) for
        $\lambda = \begin{cases} 0 \text{ if } f(\bar{\pi}) < f(\pi); \\ \Lambda \text{ Otherwise}; \end{cases}$
        If $iter > \eta$,
            The items u and v in selected move* cannot belong to the set F (i.e., move(t, u, v) is not fixed; or
            If either (t, u) or (t, v) belong to the set F,
                The move(t, u, v) is selected as the move* only when this move results in a new best solution found.
Step 5: $z = 1 - \frac{f(\pi')}{f(\pi)}$ and set $\pi = \pi'$, $f(\pi) = f(\pi')$;
        If $f(\pi) < f(\pi^*)$,
            If $iter > \eta$ and $z' = 1 - \frac{f(\pi)}{f(\pi^*)} \geq \alpha$,
                Set $F = F \cup (t,u) \cup (t,v)$;
            Set $\pi^* = \pi$ and $f(\pi^*) = f(\pi)$;

Step 6: Update $tabusize_t = f(tabusize_t)$ where
        $f(tabusize_t) = \begin{cases} tabusize_t & \text{if } z \leq 0 \\ LB_t + \frac{UB_t - LB_t}{\alpha} * z & \text{if } 0 < z \leq \alpha \\ UB_t & \text{if } \alpha < z < \beta \\ 3 * UB_t & \text{if } z \geq \beta \end{cases}$.
        Update tabu list as $tabu[t][u][v] = iter + tabusize_t$;

Step 7: Stopping Criteria: If heuristic running time < TRT, go to Step 4.
        Else, terminate the heuristic and return solution $\pi^*$ and total cost $f(\pi^*)$;
```

CHAPTER 5

COMPUTATIONAL RESULTS

5.1 Data Sets

Two test data sets are created in this research in order to test the proposed heuristics. First, a set of test problems were generated by randomly assigning items to locations in multiple periods. Then, 24 problems were selected as the test problems in data set I based on the problem factors. More specifically, data set I is randomly generated based on the following factors: the number of items *R*, number of periods *T* and number of locations *L*. In addition, there are *L*/2 work space locations and *L*/2 storage space locations, and the distance between any two locations *i* and *j* are defined in the distance matrix *D*. The number of items (resources) to be moved (*NORM*) is uniformly distributed in the interval [1, *R*] at period *t*; the number of destination locations (*NODL*) is uniformly distributed in the interval [1, *NORM*], and the number of items in destination location is uniformly distributed in the interval [1, *Cap*], where *Cap* is the capacity of the locations. In the first data set, the number of items *R* is 9, the number of locations *L* is 6, and the number of periods *T* is from 3 to 8 as shown in **Table 5.1**.

Pr. #	***R***	***L***	***T***
P01-P04	9	6	3
P05-P08	9	6	4
P09-P12	9	6	5
P13-P16	9	6	6
P17-P20	9	6	7
P21-P24	9	6	8

Table 5.1: Problem sizes for data set I.

Data set II was obtained by solving the dynamic space allocation problem (DSAP) instances presented in **McKendall *et al.* (2005)**. The DSAP assigns activities and their required items (resources) to workspaces and idle items to storage spaces with respect to minimizing the sum of the distances the items travel. Therefore, the output of the DSAP can be used as input data for the CSP. This data set has 96 test problems. The sizes of the instances in this data set are shown in **Table 5.2**.

Pr. #	***R***	***L***	***T***	***Group***
P01-P08	9	6	10	**1**
P09-P16	9	6	15	
P17-P24	9	6	20	
P25-P32	18	12	10	**2**
P33-P40	18	12	15	
P41-P48	18	12	20	
P49-P56	30	20	10	**3**
P57-P64	30	20	15	
P65-P72	30	20	20	
P73_P80	48	32	10	**4**
P81-P88	48	32	15	
P89-P96	48	32	20	

Table 5.2: Problem sizes for data set II.

5.2 Parameters Setting

The heuristic parameters were set experimentally and some use a formula.

- For the SA heuristic, the initial temperature T_0 was determined by using the formula $\exp(-[f(\pi') - f(\pi)]/T_0) = 0.25$ and $f(\pi') - f(\pi) = 0.1 * f(\pi)$, where 0.10 and 0.25 were obtained experimentally. In other words, $T_0 = -0.10 * f(\pi)/\ln(0.25)$ and the initial probability of accepting a non-improving solution, which is 10% worse then solution π, is 25%. In addition, ε, γ, T_f, and H_0 were set to 0.95, 0.05, 0.1, and number of periods (T), respectively.

- For HAS I & II heuristics, the following parameters setting are used. The numbers of ants (M) are set to 5, 10, 12, and 15 for the first 24 problems, second 24 problems, third 24 problems, and the fourth 24 problems, respectively. $E = R * T, \alpha_1 = 0.1$, and $\alpha_2 = Q = T * 10^3$, $q = 0.7$, $S = R * T$. Furthermore, the settings for the SA heuristic embedded within HAS II are $T_0 = -0.10 * f(\pi) / \ln(0.25)$, as well as ε, γ, T_f, and H_0 were set to 0.95, 0.04, 0.1, and number of periods (T), respectively.
- For the basic TS and PTS heuristics, $tabusize_t$ is set to $n_t * T / 2$. Moreover, $p = 0.4$ and $G = 10$ are the parameters setting for the PTS heuristic. The intensification strategy parameters α, η and the diversification parameters Λ, α, β are set by a statistical technique. More specifically, for each CSP problem, a certain numbers of initial solutions were generated randomly. Then, for each of these solutions, a steepest descend heuristic is used to improve the solution until no further improvement can be found (i.e., local optimum is reached). Next , the statistical information for these improved solutions are gathered, such as average percent improvement at each iteration (*API*), average iterations before reaching the local optimum (*AI*), average maximal percent improvement at each iteration (*AMPI*), average median objective function value (*AMOFV*), and average minimal objective function value (*AOFV*). Then, the parameters α is set to *API*, η is equal to *AI*, Λ is set to *AMOFV/AOFV* and β is set to *AMPI*. In addition, LB_t and UB_t are defined as:

$$LB_t = n_t * T / 3 \ \ \forall t > 1 \ \text{ and } \ UB_t = n_t * T * 2 \ \ \forall t > 1.$$

5.3 Test Environment

In all experiments, a Pentium IV 2.4GHz computer with 1 Giga-byte of memory was used to solve the CSP instances from data set I and II using the proposed heuristics. The heuristics were coded using the C++ programming language under Windows XP operating system. In order to make fair comparisons, the proposed SA, HAS and TS heuristics ran for the same amount of computational time for each CSP problem instance.

5.4 Experimental Results

a) Data Set I

For data set I, the first construction algorithm is used to generate initial solutions for the heuristics. SA, TS, PTS, TS with strategies (TS/S), and HAS I/II are applied to this data set. All proposed heuristics ran only once and ran the same amount of time for each test problem.

Table 5.3 shows the results obtained from the proposed heuristics. The optimal solutions were obtained for all of the test problems in this set and were obtained using the mathematical model presented in section 4.2.1 and CPLEX version 6.6. The following CPLEX parameter settings were used: strong branching for variable selection, depth first for node selection, dual simplex for start algorithm, and the rest were default settings. Also, their computational times are listed. The bold numbers represent the optimal OFVs. All times in the tables are given in minutes. As a result, all of the proposed heuristics obtained the optimal solutions for all 48 test problems. Notice the run times for the heuristics are much less than the run time for the mathematical model with CPLEX. The main reason for using this data set was to test the mathematical model.

Pb #	Optimal Solution		SA	TS	PTS	TS/S	HASI/II	Heuristic Time
	Final	Time						
P01	**57**	0.01	**57**	**57**	**57**	**57**	**57**	0.00
P02	**83**	0.05	**83**	**83**	**83**	**83**	**83**	0.00
P03	**62**	0.03	**62**	**62**	**62**	**62**	**62**	0.00
P04	**88**	0.11	**88**	**88**	**88**	**88**	**88**	0.01
P05	**65**	0.08	**65**	**65**	**65**	**65**	**65**	0.01
P06	**119**	7.60	**119**	**119**	**119**	**119**	**119**	0.01
P07	**72**	0.02	**72**	**72**	**72**	**72**	**72**	0.01
P08	**91**	0.23	**91**	**91**	**91**	**91**	**91**	0.02
P09	**121**	0.33	**121**	**121**	**121**	**121**	**121**	0.02
P10	**150**	19.42	**150**	**150**	**150**	**150**	**150**	0.02
P11	**118**	2.02	**118**	**118**	**118**	**118**	**118**	0.02
P12	**150**	19.53	**150**	**150**	**150**	**150**	**150**	0.01
P13	**104**	0.10	**104**	**104**	**104**	**104**	**104**	0.01
P14	**189**	1527.36	**189**	**189**	**189**	**189**	**189**	0.02
P15	**114**	0.12	**114**	**114**	**114**	**114**	**114**	0.02
P16	**229**	8.48	**229**	**229**	**229**	**229**	**229**	0.02
P17	**145**	7.05	**145**	**145**	**145**	**145**	**145**	0.02
P18	**278**	1947.32	**278**	**278**	**278**	**278**	**278**	0.02
P19	**157**	60.97	**157**	**157**	**157**	**157**	**157**	0.02
P20	**246**	1014.73	**246**	**246**	**246**	**246**	**246**	0.03
P21	**169**	12.93	**169**	**169**	**169**	**169**	**169**	0.03
P22	**320**	2915.87	**320**	**320**	**320**	**320**	**320**	0.04
P23	**197**	318.12	**197**	**197**	**197**	**197**	**197**	0.04
P24	**347**	2755.59	**347**	**347**	**347**	**347**	**347**	0.03

Table 5.3: Computational results for generated data set I.

b) Data Set II

The second data is used to further verify the speed and robustness of the proposed heuristics. This data set has test problems with 9-48 items, 6-32 locations and 10, 15 and

20 periods as shown in **Table 5.2**. Compared to data set I, data set II is much larger and has much larger problems, which cannot be solved by using exact methods (i.e., the proposed mathematical model and CPLEX) in reasonable time. The TS and TS/S heuristics, which are deterministic techniques, are run once with each of the solutions obtained using the construction algorithms for each test problem. SA, HAS I and II, PTS are run 5 times with each of the solutions obtained from the construction algorithms. Data set II is divided into 4 groups, each of which has 24 test problems. From group 1 to group 4, the sizes of the test problems are increased. The test problems in group 1 are relative small, groups 2 and 3 have medium size test problems, and the test problems in group 4 are the largest.

No.	**TS**	**PTS**	**TS/S**	**HASI**	**HASII**	**SA**
P01-P24	22	24	24	23	22	23
P25-P48	5	12	15	12	12	13
P49-P72	1	10	12	6	12	3
P73-P96	0	5	12	4	11	4
Total	**28**	**51**	**63**	**45**	**57**	**43**
Percent	**29.2%**	**53.1%**	**65.6%**	**46.9%**	**59.4%**	**44.8%**

Table 5.4: # of best solutions found obtained by the proposed heuristics.

No.	**TS**	**PTS**	**TS/S**	**HASI**	**HASII**	**SA**
P01-P24	0.22%	0.00%	0.00%	0.09%	0.12%	0.09%
P25-P48	2.57%	0.43%	0.27%	0.47%	0.36%	0.44%
P49-P72	3.51%	0.38%	0.14%	0.37%	0.26%	1.40%
P73-P96	3.69%	0.27%	0.12%	0.31%	0.19%	1.15%
Average	2.49%	0.27%	0.13%	0.31%	0.23%	0.77%

Table 5.5: Average percent each heuristic is away from the best found solutions.

Table 5.4 shows the number of best solutions obtained by each of the proposed heuristics. For the first group (i.e., test problems 01-24), the test problems have 9 resources, 6 locations, and 3 – 5 periods, which are the smaller size problems. All proposed heuristics performed well for this group of test problems. More specifically, TS obtained the best found solution 22 times, PTS 24 times, TS/S 24 times, HAS I 23 times, HAS II, 22 times, and SA 23 times. The detail results for the first group are given in **Table 5.6**. Note, the bolded OFVs are the best found OFVs. **Table 5.5** gives the average percent each heuristic is away from best found solutions.

However, the proposed heuristics perform differently for the other 3 groups of test problems. More specifically, TS obtained the best found solutions 5, 1, 0 time(s), PTS obtained 12, 10, 5 times, TS/S obtained 15, 12, 12 times, HAS I obtained 12, 6, 4 times, HAS II obtained 12, 12, 11 times, and SA obtained 13, 3 and 4 times. See the results for each test problem in groups 2, 3, and 4 in **Tables 5.7**, **5.8**, and **5.9**, respectively. Clearly, TS/S outperformed all other heuristics since it obtained 39 best solutions of 72 test problems, which is the most. In contrast, simple TS only found 6 best solutions. However, HAS II obtained 12, 12 and 11 times (total of 35 out of 72 test problems) for these 3 groups of 24 test problems, which has the most stable performance for data set II. PTS performed well for medium size problems, but relatively worse for the last group of test problems. Moreover, HAS I performed slightly better than SA since HAS I obtained best found 22 times and SA obtained 20 of 72 test problems.

Tables 5.6 – 5.9 list details results and computational times for each group of test problems in data set II. The bold numbers are the best found OFVs, and times are given in minutes. The HAS II heuristic is HAS I combined with SA heuristic. Although this integration increases the computational complexity, the results show that HAS II improved HAS I, since the average percents HAS II OFVs are below HAS I are -0.01%, 0.03%, 0.03% and 0.03% for groups 1, 2, 3, and 4, respectively. More importantly, HAS II obtained 42 solutions better than HAS I out of the 96 test problems (43.8%).

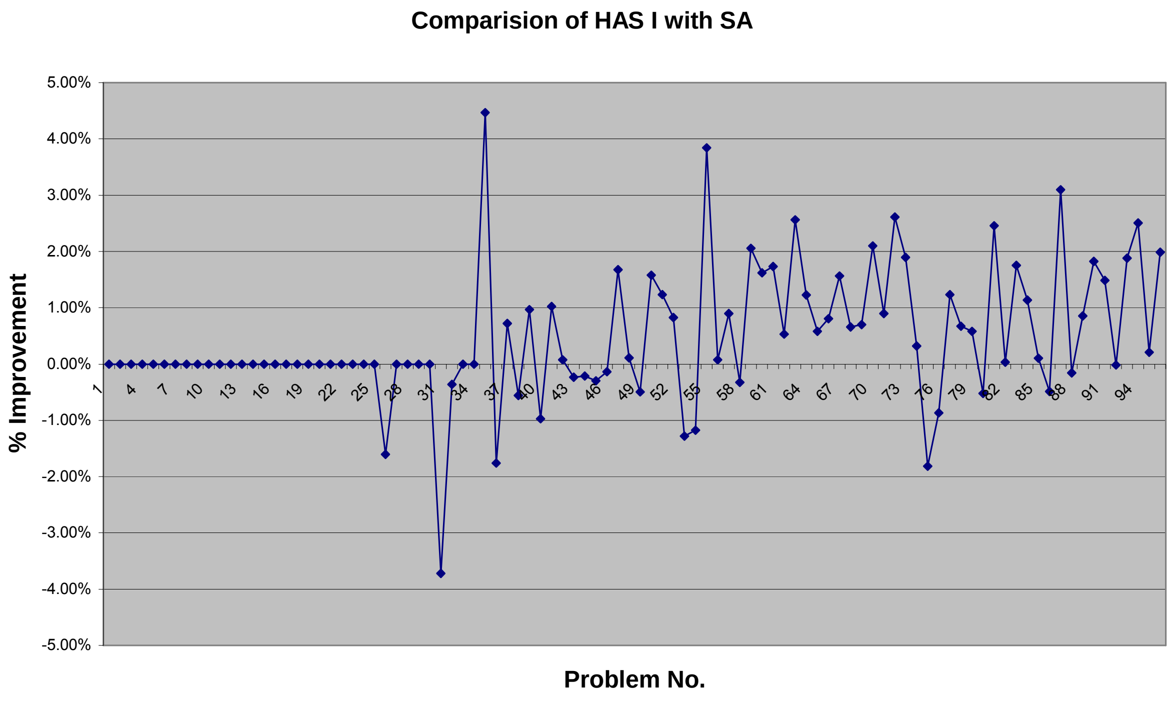

Figure 5.1: % Improvement of HAS I over SA heuristic.

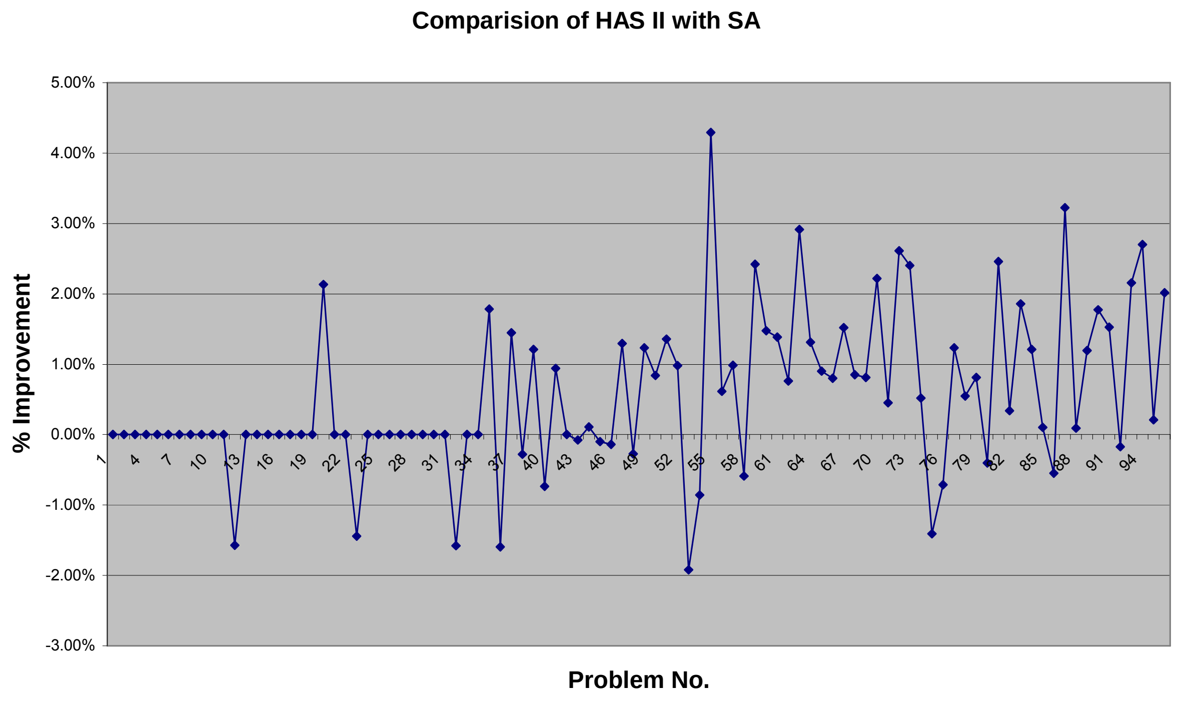

Figure 5.2: % Improvement of HAS II over SA heuristic.

When comparing the HAS I and HAS II with SA heuristic, overall, the solutions obtained from the HAS I and II heuristics are 0.46% and 0.54% improved to solutions obtained from the SA heuristic. However, when considering each of the groups of test problems separately, the percent of improvement of HAS I over SA heuristic are 0.00%, -0.03%, 1.03% and 0.84%, respectively. The percent of improvement of HAS II over SA heuristic are -0.04%, 0.08%, 1.14% and 0.96% for each of the four groups. Therefore, both the proposed HAS heuristics outperformed the SA heuristic for larger size problems, and the SA heuristic performed well for smaller size problems. See a comparisons of the HAS heuristics and SA in **Figure 5.1** and **Figure 5.2**. It shows that both the HAS I and II heuristics have a relative stable percent improvement over the SA heuristic for large size problems.

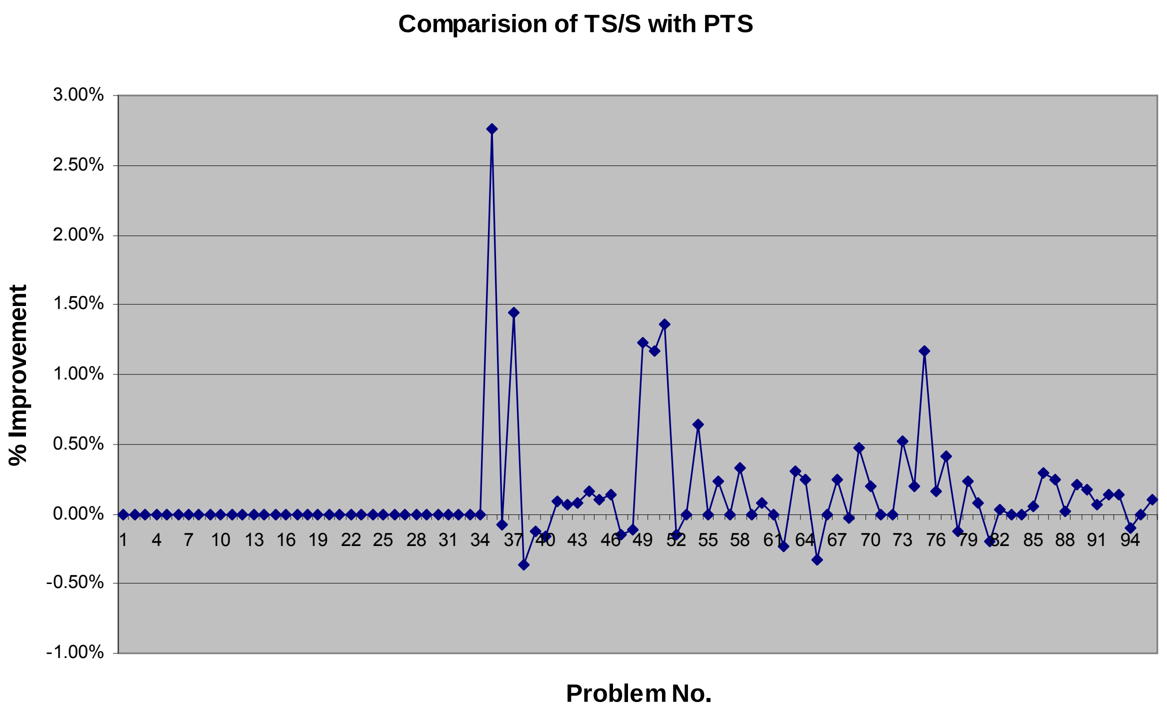

Figure 5.3: % Improvement of TS/S heuristic over PTS heuristic.

For the proposed TS heuristics, based on the previous analyses, it is obvious that both proposed PTS and TS/S heuristic outperformed the basic TS heuristic. See a comparisons of TS/S and PTS in **Figure 5.3**, which shows the percent improvement of TS/S heuristic over PTS heuristic for the CSP. More specifically, when comparing the TS/S heuristic with PTS heuristic, TS/S heuristic obtained better solutions than PTS 38 out of 96 test problems, and PTS heuristic outperformed TS/S heuristic 13 times. In addition, they obtained the same results 45 times, which include all test problems from P01 to P34. Therefore, TS/S heuristic performed better than PTS heuristic for medium and large size test problems. Also, it is important to note that PTS heuristic ran 15 times for each test problem (i.e., 5 runs for each of the 3 solutions obtained from the construction algorithms) and TS/S heuristic only ran 3 times for each problem (i.e., single run for each of 3 solutions generated from the construction algorithms). Therefore, TS/S heuristic total run time was 1/5 of the PTS total run time.

Moreover, computational experiments show that the solution qualities of the TS heuristic really depend on the qualities of the initial solutions. When comparing the construction algorithms, CAI, CAII, and CAIII obtained the better solution 2, 69, 66 times, respectively. When only consider the solutions from the basic TS with CAI, CAII, and CAIII construction algorithms, TS/CAI performed best 32 times, TS/CAII 76 times, and TS/CAIII 73 times. Because the qualities of the constructed solutions by using CAII and CAIII are relative the same, both of them are much better than the solutions constructed by CAI. It is clear that the initial solution quality affects the performance of the TS heuristic. However, the performances of the other tabu search heuristics (i.e., PTS and TS/S) were not depended on the quality of the initial solutions. It is obvious that the performances of the stochastic heuristics (i.e., HAS I, HAS II, and SA) are not depended on the quality of the initial solutions. Therefore, when applying these heuristics to the CSP, more emphasis should be place on the heuristics themselves instead of the construction algorithms.

No	**Initial Cost**	**TS**				**HAS**			**SA**	**Best of All**	**Time**
		TS	**PTS**	**TS/S**	**Best**	**HASI**	**HASII**	**Best**			
1	**205**	**205**	**205**	**205**	**205**	**205**	**205**	**205**	**205**	***205***	0.03
2	359	**339**	**339**	**339**	**339**	**339**	**339**	**339**	**339**	***339***	0.05
3	243	**233**	**233**	**233**	**233**	**233**	**233**	**233**	**233**	***233***	0.03
4	324	**314**	**314**	**314**	**314**	**314**	**314**	**314**	**314**	***314***	0.05
5	**213**	**213**	**213**	**213**	**213**	**213**	**213**	**213**	**213**	***213***	0.03
6	385	**375**	**375**	**375**	**375**	**375**	**375**	**375**	**375**	***375***	0.05
7	274	**254**	**254**	**254**	**254**	**254**	**254**	**254**	**254**	***254***	0.05
8	369	**346**	**346**	**346**	**346**	**346**	**346**	**346**	**346**	***346***	0.03
9	410	**390**	**390**	**390**	**390**	**390**	**390**	**390**	**390**	***390***	0.07
10	655	**645**	**645**	**645**	**645**	**645**	**645**	**645**	**645**	***645***	0.07
11	462	**442**	**442**	**442**	**442**	**442**	**442**	**442**	**442**	***442***	0.07
12	720	657	**637**	**637**	**637**	**637**	647	**637**	**637**	***637***	0.07
13	403	**373**	**373**	**373**	**373**	**373**	**373**	**373**	**373**	***373***	0.05
14	595	**585**	**585**	**585**	**585**	**585**	**585**	**585**	**585**	***585***	0.05
15	566	**460**	**460**	**460**	**460**	**460**	**460**	**460**	**460**	***460***	0.07
16	733	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	***670***	0.07
17	**467**	**467**	**467**	**467**	**467**	**467**	**467**	**467**	**467**	***467***	0.08
18	856	**806**	**806**	**806**	**806**	**806**	**806**	**806**	**806**	***806***	0.08
19	651	**621**	**621**	**621**	**621**	**621**	**621**	**621**	**621**	***621***	0.10
20	1031	938	**918**	**918**	**918**	938	**918**	**918**	938	***918***	0.08
21	711	**615**	**615**	**615**	**615**	**615**	**615**	**615**	**615**	***615***	0.08
22	926	**853**	**853**	**853**	**853**	**853**	**853**	**853**	**853**	***853***	0.08
23	848	**692**	**692**	**692**	**692**	**692**	702	**692**	**692**	***692***	0.10
24	981	**786**	**786**	**786**	**786**	**786**	**786**	**786**	**786**	***786***	0.08

Table 5.6: Computational results for 1st group in data set 2

No	Initial Cost	TS				HAS			SA	Best of All	Time
		TS	PTS	TS/S	Best	HASI	HASII	Best			
25	610	**490**	**490**	**490**	**490**	**490**	**490**	**490**	**490**	***490***	0.45
26	813	**622**	**622**	**622**	**622**	632	**622**	**622**	**622**	***622***	0.43
27	982	533	**513**	**513**	**513**	**513**	**513**	**513**	**513**	***513***	0.50
28	1038	**730**	**730**	**730**	**730**	**730**	**730**	**730**	**730**	***730***	0.47
29	544	**441**	**441**	**441**	**441**	**441**	**441**	**441**	**441**	***441***	0.43
30	783	675	**660**	**660**	**660**	**660**	**660**	**660**	**660**	***660***	0.45
31	791	572	**537**	**537**	**537**	557	**537**	**537**	**537**	***537***	0.48
32	1167	**824**	**824**	**824**	**824**	827	837	827	**824**	***824***	0.48
33	1107	834	**794**	**794**	**794**	**794**	**794**	**794**	**794**	***794***	0.70
34	1434	1075	**1065**	**1065**	**1065**	**1065**	**1065**	**1065**	**1065**	***1065***	0.70
35	1249	926	906	881	881	**856**	880	**856**	896	***856***	0.77
36	1670	1266	1213	1214	1213	1214	1212	1212	**1193**	***1193***	0.73
37	895	702	692	**682**	**682**	687	**682**	**682**	692	***682***	0.68
38	1312	1099	**1079**	1083	**1079**	1085	1082	1082	**1079**	***1079***	0.70
39	1276	827	**817**	818	**817**	819	**817**	**817**	827	***817***	0.75
40	1793	1262	1236	1238	1236	1241	1238	1238	**1229**	***1229***	0.75
41	1587	1171	1161	1160	1160	**1159**	1160	**1159**	1171	***1159***	0.93
42	1844	1429	1409	**1408**	**1408**	**1408**	1409	**1408**	1409	***1408***	0.95
43	1765	1379	1269	**1268**	**1268**	1275	1273	1273	1272	***1268***	1.05
44	2566	1920	1864	1861	1861	1864	**1858**	**1858**	1860	***1858***	1.03
45	1326	1023	1013	**1012**	**1012**	1016	1014	1014	1013	***1012***	0.90
46	1798	1488	1455	**1453**	**1453**	1457	1457	1457	1455	***1453***	0.93
47	1958	1346	1293	1295	1293	**1291**	1296	**1291**	1313	***1291***	1.05
48	2562	1840	1830	1832	1830	**1828**	1835	**1828**	1830	***1828***	1.02

Table 5.7: Computational results for 2nd group in data set 2

No	**Initial Cost**	**TS**				**HAS**			**SA**	**Best of All**	**Time**
		TS	**PTS**	**TS/S**	**Best**	**HASI**	**HASII**	**Best**			
49	1085	850	812	**802**	**802**	816	**802**	**802**	812	***802***	2.82
50	1210	966	946	**935**	**935**	936	943	936	951	***935***	2.87
51	1162	819	809	**798**	**798**	799	**798**	**798**	809	***798***	3.12
52	1892	**1315**	**1315**	1317	**1315**	1317	**1315**	**1315**	1328	***1315***	3.05
53	1164	821	791	791	791	791	796	791	**781**	***781***	2.78
54	1351	975	942	936	936	943	940	940	**932**	***932***	2.83
55	1404	885	**847**	**847**	**847**	851	**847**	**847**	885	***847***	3.15
56	2002	1305	1302	1299	1299	1304	**1297**	**1297**	1305	***1297***	3.12
57	1576	1127	**1106**	**1106**	**1106**	1107	**1106**	**1106**	1117	***1106***	4.28
58	2047	1545	1535	1530	1530	1530	1534	1530	**1525**	***1525***	4.30
59	2166	1406	**1330**	**1330**	**1330**	1335	**1330**	**1330**	1363	***1330***	4.88
60	2166	1406	1336	1335	1335	**1334**	1336	**1334**	1356	***1334***	4.82
61	1677	1184	**1134**	**1134**	**1134**	**1134**	1138	**1134**	1154	***1134***	4.43
62	2416	1768	**1689**	1693	**1689**	1697	1693	1693	1706	***1689***	4.38
63	2494	1720	1637	**1632**	**1632**	1638	**1632**	**1632**	1681	***1632***	5.03
64	3775	2510	2411	**2405**	**2405**	2411	2409	2409	2441	***2405***	4.97
65	2265	1603	**1533**	1538	**1533**	1544	1539	1539	1553	***1533***	5.90
66	3094	2289	**2224**	**2224**	**2224**	**2224**	**2224**	**2224**	2242	***2224***	5.90
67	3025	2122	2082	2077	2077	**2074**	2075	**2074**	2107	***2074***	6.80
68	4364	3075	**3025**	3026	**3025**	3031	**3025**	**3025**	3051	***3025***	6.70
69	2728	1760	1710	**1702**	**1702**	1708	1706	1706	1720	***1702***	6.22
70	3646	2586	2477	2472	2472	2472	**2469**	**2469**	2525	***2469***	6.20
71	3531	2540	2427	2427	2427	**2425**	2436	**2425**	2447	***2425***	7.18
72	5223	3672	**3547**	**3547**	**3547**	**3547**	**3547**	**3547**	3642	***3547***	7.15

Table 5.8: Computational results for 3rd group in data set 2

No	**Initial Cost**	**TS**				**HAS**			**SA**	**Best of All**	**Time**
		TS	**PTS**	**TS/S**	**Best**	**HASI**	**HASII**	**Best**			
73	2022	1393	1353	1346	1346	1347	**1340**	**1340**	1373	***1340***	17.28
74	2101	1573	1543	**1540**	**1540**	1543	**1540**	**1540**	1548	***1540***	17.23
75	2640	1790	1720	**1700**	**1700**	1736	1729	1729	1705	***1700***	18.93
76	3636	2586	2558	2554	2554	2560	2556	2556	**2538**	***2538***	18.48
77	2024	1255	1205	**1200**	**1200**	**1200**	**1200**	**1200**	1215	***1200***	16.22
78	2436	1675	1635	1637	1635	**1634**	1636	**1634**	1645	***1634***	16.25
79	3001	1790	1710	**1706**	**1706**	1710	**1706**	**1706**	1720	***1706***	18.52
80	4323	2657	2492	2490	2490	2496	2493	2493	**2483**	***2483***	18.57
81	2933	2064	**1974**	1978	**1974**	1984	1984	1984	2034	***1974***	27.50
82	4445	3022	2932	2931	2931	2936	**2927**	**2927**	2937	***2927***	27.17
83	4924	3062	**2907**	**2907**	**2907**	2910	**2907**	**2907**	2962	***2907***	30.27
84	6585	4169	**4004**	**4004**	**4004**	4008	4005	4005	4054	***4004***	29.85
85	3241	2063	1978	1977	1977	**1976**	**1976**	**1976**	1978	***1976***	26.78
86	4787	3162	3112	3103	3103	3102	3104	3102	**3087**	***3087***	26.97
87	5082	3154	2917	**2910**	**2910**	2914	**2910**	**2910**	3007	***2910***	30.62
88	6896	4491	4441	**4440**	**4440**	4451	**4440**	**4440**	4444	***4440***	30.77
89	4387	3002	2909	2903	2903	2911	**2901**	**2901**	2936	***2901***	36.49
90	6202	4053	3946	3939	3939	**3937**	3939	**3937**	4010	***3937***	39.44
91	6883	4490	4445	**4442**	**4442**	4450	4448	4448	4517	***4442***	41.63
92	9285	6107	5916	5908	5908	5903	5912	5903	**5902**	***5902***	44.98
93	4866	3013	2865	**2861**	**2861**	2870	2862	2862	2925	***2861***	40.66
94	5982	4086	**3960**	3964	**3960**	3968	**3960**	**3960**	4070	***3960***	41.78
95	7741	4967	**4764**	**4764**	**4764**	4767	4767	4767	4777	***4764***	49.56
96	10542	6706	6526	**6519**	**6519**	6523	6521	6521	6655	***6519***	48.17

Table 5.9: Computational results for 4th group in data set 2

CHAPTER 6

CONCLUSIONS

6.1Summary of Research

In industry, rearranging the items (or resources) from existing positions to new locations may be a costly operation and may represent a significantly portion of the overall project budget. Cranes are the most popular material handling equipment for relocating items. However, such items are bulky, and moving these items with a single crane becomes difficult. Hence, it is essential to develop a good crane work route to ensure high efficiency and lower cost. In this research, this problem was defined as the CSP. More specifically, the CSP minimizes the total crane travel cost and loading/unloading cost for the crane to relocate items in a multiple time period horizon. The CSP is related to the well-known TSP and a variant of the TSP, SCP. A binary integer mathematical model was formulated for solving small CSPs. Since solving large size CSPs is computational intractable, 3 construction algorithms, SA, HAS I, HAS II, TS, PTS, and TS/S heuristics are developed to provide crane routes for larger problems in reasonable computation time.

6.2 Recommendations for Future Research

The following recommendations may be considered for future research:

1. Integrate the CSP with its related problem, the dynamic space allocation problem, which may produce better solutions.

2. Consider length, width, height, and stackability of items as well as the volume of the locations when considering capacity of locations.
3. Consider using multiple cranes.
4. Combine other existing heuristics (e.g., TS and HAS) which may obtain better solutions for the CSP.
5. Consider solving the CSP for other applications.

REFERENCES

Anily, S. and Bramel, J., Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries, *Naval Research Logistics*, v 46, n 6, Sep, 1999, p 654-670

Anily, S. and Hassin, R., The Swapping Problem, *Networks*, v22, 1992, p 419- 433

Anily, S. and Mosheiov, G., Traveling salesman problem with delivery and backhauls, *Operations Research Letters*, v 16, n 1, Aug, 1994, p 11-18

Anily, S., Gendreau, M. and Laporte, G., Swapping problem on a line, *SIAM Journal on Computing*, v 29, n 1, Sep, 1999, p 327-335

Ball, M., Magnanti, T., Monma, C. and Nemhauser, G., Network Routing, *Handbooks in Operations Research and Management Science*, vol. 8., 1995

Bouleimen, K. and Lecocq, H., A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, *European Journal of Operational Research*, v 149, n 2, 1 Sept. 2003, p 268-81

Bramel, J. and Simchi-Levi, D., The Logics of Logistics, theory, Algorithms, and Applications for Logistics Management, 2nd edition, *Springer Series in Operations Research*, Springer-Verlag, New York, 1999

Burkard, R.E., Deineko, V.G., van Dal, R., van der Veen, J.A.A. and Woeginger, G.J., Well-solvable special cases of the traveling salesman problem: A survey, *SIAM Review*, v 40, n 3, Sep, 1998, p 496-546

Casco, D.O., Golden B.L. and Wasil, E.A., Vehicle Routing with Backhauls: Models, Algorithms and Case Studies. *Vehicle Routing: Methods and Studies*, North Holland, Amsterdam, 1988, p 127-147

Charikar, M. and Raghavachari, B., Finite capacity dial-a-ride problem, *Annual Symposium on Foundations of Computer Science - Proceedings*, 1998, p 458-467

Charikar, M., Khuller, S. and Raghavachari, B., Algorithms for capacitated vehicle routing, *SIAM Journal on Computing*, v 31, n 3, 2002, p 665-682

Chalasani, P. and Motwani, R., Approximating capacitated routing and delivery problems, *SIAM Journal on Computing*, v 28, n 6, Jun-Aug, 1999, p 2133-2149

Chopra, S. and Meindl, P., Supply Chain Management, *Prentice Hall*, 2nd edition, May 1, 2003, p 415-420

Christofides, N. and Colloff, I., The rearrangement of items in a warehouse, *Operations Research*, v 21(2), 1973, p577-589

Clarke, G. and Wright, J.W., Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research*, 12, 1964, p 568-581

Colorni, A., Dorigo, M. and Maniezzo, V., Distributed optimization by ants colonies, In: Varela, F, Bourgine, P (Eds.), *First European Conference on Artificial Life*, 1991. p. 134-42.

Crainic, T.G. and Laporte, G., Fleet Management and Logistics, *Centre for Research on Transportation, 25th Anniversary Series, Kluwer Academic Publishers*, 1999

Current, J., Multiple objectives in transportation network design and routing, *European Journal of Operational Research*, v 65, n 1, Feb 19, 1993, p 1-3

Dantzig, G.B. and Ramser, J.H., The truck dispatching problem, *Management Science*, 6, 1959, p 80-91

Dorigo, M. and Gambardella, L.M., Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, v 1, n 1, April 1997, p 53-66

Dorigo, M., Maniezzo, V. and Colorni, A., The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics—PartB*, 1996, 26(1), 29-41.

Faggioli, E. and Bentivoglio, C.A., Heuristic and exact methods for the cutting sequencing problem, *European Journal of Operational Research*, v 110, n 3, Nov 1, 1998, p 564-575

Frederickson, G.N. and Guan, D.J., Preemptive ensemble motion planning on a tree, *SIAM J. Computing*, 21, 1992, no. 6, 1130--1152

Frederickson, G.N., Hecht, M.S. and Kim, C.E., Approximation algorithms for some routing problems, *SIAM J. Computing*, 7, 1978, p 178-193

Gambardella, L.M., Taillard, E.D. and Dorigo, M., Ant colonies for the quadratic assignment problem, *Journal of Operations Research Society*, 1999, v50: p 167-76.

Garey, M.R. and Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, *W.H. Freeman and Company*, 1979.

Gendreau, M., Laporte, G. and Hertz, A., Approximation algorithm for the Traveling Salesman Problem with Backhauls, *Operations Research*, v 45, n 4, Jul-Aug, 1997, p

639-641

Gendreau, M., Hertz, A. and Laporte, G., Traveling salesman problem with backhauls, *Computers & Operations Research*, v 23, n 5, May, 1996, p 501-508

Gendreau, M. and Potvin, J., Metaheuristics in Combinatorial Optimization, *Annals of Operations Research*, n 140, 2005, p 189 - 213

Groetschel, M. and Holland, O., Solution of large-scale symmetric travelling salesman problems, *Mathematical Programming*, v 51, n 2, Sep, 1991, p 141-202

Henderson, D., Vaughan, D.E. Jacobson, S.H., Wakefield, R.R. and Sewell, E.C., Solving the shortest route cut and fill problem using simulated annealing. *European Journal of Operational Research*, v 145, n 1, 2003, p72-84

Hernandez-Perez, H. and Salazar-Gonzalez, J., Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, *Transportation Science*, v 38, n 2, May, 2004, p 245-255

Hernandez-Perez, H. and Salazar-Gonzalez, J., A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, *Discrete Applied Mathematics*, v 145, n 1, 2004, SPEC. ISS., p126-139

Hunsaker, B. and Savelsbergh, M., Efficient feasibility testing for dial-a-ride problems, *Operations Research Letters*, v 30, n 3, June, 2002, p 169-173

Kirpatrick S., Gelatt, C.D. and Vecchi, M.P., Optimization by Simulated Annealing, *Science*, 220(4598), 1983, p 671-680

Kolisch, R., Serial and parallel resource-constrained project scheduling methods revisited:

theory and computation, European Journal of Operational Research, v 90, n 2, April 19, 1996, p 320-333

Kolisch, R. and Hartmann, S., Experimental Investigation of Heuristics for Resource-Constrained project scheduling: An update, 2004 (downloadable from website: http://129.187.106.231/psplib/files/).

Laporte, G., Traveling salesman problem: An overview of exact and approximate algorithms, *European Journal of Operational Research*, v 59, n 2, Jun 10, 1992, p 231-247

Laporte, G. and Osman, I.H., Routing problems: a bibliography, *Annals of Operations Research*, v 61, Dec. 1995, p 227-62

Lim, A., Rodrigues, B., Xiao, F. and Zhu, Y., Crane scheduling using Tabu search, *Proceedings of the International Conference on Tools with Artificial Intelligence*, 2002, p 146-153

Lim, A., Rodrigues, B. and Zhang, J., Tabu Search embedded simulated annealing for the shortest route cut and fill problem, *Journal of the Operational Research Society*, 2004, p 1-9

Maniezzo, V. and Colorni, A., The ant system applied to the quadratic assignment problem, IEEE Trans Know, *Data Engineering*, 1999, 11 p 769-78

McKendall Jr., A.R., Noble, J.S. and Klein, C.M., Simulated annealing heuristics for managing resources during planned outages at electric power plants, *Computers and Operations Research*, v 32, n 1, January, 2005, p 107-125

McKendall Jr., A.R., Shang, J., Noble, J.S. and Klein, C.M., A Simulated Annealing

Heuristic for a Crane Sequencing Problem, accepted for publication in International Journal of Industrial Engineering, 2006

McKendall Jr., A.R. and Shang, J., Hybrid Ant Systems for the Dynamic Facility Layout Problem, *Computers and Operations Research*, v 33, n 3, March, 2006, p 790-803

Miyamoto, T., Nakatyou, K. and Kumagai, S., Route planning method for a dial-a-ride problem, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, v 4, 2003, p 4002-4007

Mosheiov, G., The traveling Salesman Problem with Pick-up and Delivery, *European Journal of Operational Research*, v 79, 1994, p 299-310

Psaraftis, H.N., Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem, *Transportation Science*, v 14, n 2, May, 1980, p 130-154

Renaud, J., Boctor, F. and Ouenniche, J., Heuristic for the Pickup and Delivery Traveling Salesman Problem, *Computers and Operations Research*, v 27, n 9, Aug, 2000, p 905-916

Renaud, J., Boctor, F. and Laporte, G, Perturbation heuristics for the pickup and delivery traveling salesman problem, *Computers and Operations Research*, v 29, n 9, August, 2002, p 1129-1141

Ruiz, R., Maroto, C. and Alcaraz, J., A decision support system for a real vehicle routing problem, *European Journal of Operational Research*, v 153, n 3 SPEC. ISS., Mar 16 2003, 2004, p 593-606

Smith, K., Palaniswami, M. and Krishnamoorthy, M., Traditional heuristic versus

Hopfield neural network approaches to a car sequencing problem, *European Journal of Operational Research*, v 93, n 2, Sep 6, 1996, p 300-316.

Stutzle, T. and Dorigo, M., ACO algorithms for the quadratic assignment problem, In: Corne, D., Dorigo, M., and Glover, F. (Eds.), *New Ideas in Optimization*. New York: McGraw-Hill, 1999. p. 33-50.

Sural, H. and Bookbinder, J.H., The single-vehicle routing problem with unrestricted backhauls, *Networks*, v 41, n 3, 2003, p 127-136

U.S. Department of Labor, Occupational Outlook Handbook: 2004-2005, p 498-499, *Jist Publishing*, May, 2004, p 498-499.

Van den Berg, J.P., Literature survey on planning and control of warehousing systems, *IIE Transactions*, v 31, n 8, Aug, 1999, p 751-762

Wen, U-P. and Yeh, C-I., Tabu search methods for the flow shop sequencing problem, *Journal of the Chinese Institute of Engineers, Transactions of the Chinese Institute of Engineers*, v 20, n 4, Jul, 1997, p 465-470.