NETWORK OPTIMIZATION APPROACHES TO SOLVE THE STOCHASTIC AND DYNAMIC FACILITY LAYOUT PROBLEMS AND REDUCE SUPPLY CHAIN COSTS

by

Gowtham Balachandran. B.E.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Industrial Engineering
May 2018

Committee Members:

Clara Novoa, Chair

Tongdan Jin

Apan Qasem

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

DEDICATION

I must express my very profound gratitude to my parents, and to my friends and family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

ACKNOWLEDGEMENTS

First, I would like to thank my thesis advisor **Dr. Clara Novoa**, Ph.D., Associate Professor in the Ingram School of Engineering at Texas State University. The way she worked with me was exceptional. The time she spent in clearing all my doubts was the way it helped me to complete this thesis. The door to Dr. Novoa's office was always open whenever I had a question about my research or about writing this document. She consistently allowed this paper to be my own work but steered me in the right direction whenever she thought I needed it. I also would like to thank the professors who were involved in the experimental phase of this research project: **Dr. Tongdan Jin,** Ph.D., Associate Professor in the Ingram School of Engineering and **Dr. Apan Qasem**, Ph.D., Associate Professor in the Department of Computer Science at Texas State University. The passionate participation and input of Dr. Qasem and Dr. Jin was very helpful on the successful development of this project. I also like to acknowledge **Dr. Vishu Viswanathan,** Ph.D., Ingram Professor in the Ingram School of Engineering at Texas State University as the second reader of this thesis. I am gratefully indebted for his valuable comments. I would like to thank Dr. Jaydeep Balakrishnan, Professor at the Operations and Supply Chain Management area at the Haskayne School of Business at the University of Calgary and director of the Canadian Centre for Advanced Supply Chain Management and Logistics for providing to my advisor the datasets for experimenting with the Dynamic Facility Layout Problem (DFLP).

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Description
ACO	Ant Colony Optimization
AHP	Analytic Hierarchy Process
AIS	Artificial Immune System
ANN	Artificial Neural Networks
BFS	Basic Feasible Solution
BSA	Backtracking Search Algorithm
CNC	Computer Numerical Control
CONGA	Conway and Venkataraman's Genetic Algorithm
CPM	Cutting Plane Method
CRAFT	Computerized Relative Allocation of Facilities Technique
CSA	Chaotic Simulated Annealing
CSP	Constrained Shortest Path
DEA	Data envelopment analysis
DFLP	Dynamic Facility Layout Problem
DP	Dynamic Programming
DWLP	Dynamic Warehouse Location Programming
ES	Expert System
FA	Firefly Algorithm
FLP	Facility Layout Problem
FS	Fuzzy System
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
GT	Graph Theoretic models
HAS	Hybrid Ant Systems
IP	Integer Programming
IRP	Interpretive Ranking Process
LP	Linear Programming
LNM	Linear network model
LNP	Linear network problem
MADM	Multiple attribute decision making
MCNFP	Minimum-Cost Network Flow Problem
MHC	Material Handling Cost
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MTEL	Minimize the Total Expected Loss
NLGA	Nested Loop Genetic Algorithm
NLIP	Non-Linear Integer Programming
NLP	Non-Linear Program

NP hard	Non-Deterministic Polynomial Time hard
NSA	Network Simple Algorithm
NSM	Network Simplex Method
PSP	Parallel Shortest Path
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem
QSP	Quadratic Set Covering
RC	Relocation Cost
SA	Simulated Annealing
SDFLP	Stochastic Dynamic Facility Layout Problem
SFLP	Static Facility Layout Problem
SPP	Shortest Path Problem
TACC	Texas Advanced Computing Center
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TS	Tabu Search

ABSTRACT

This thesis researches on the Dynamic Facility Layout Problem (DFLP) and the Stochastic and Dynamic Facility Layout Problem (SDFLP). The problems are extensions to the static or single-period facility layout problem (SFLP). They assume that there are fluctuations in the products' demands and consequently in the flows of material (and/or final products) between facilities in a given planning horizon. Fluctuations in flows of material are also due to the introduction of new products, disasters, and other production and marketing changes impacting the supply chain. In the DFLP, the flows of material between facilities vary over time but they are assumed known. In the SDFLP, the flows between facilities are uncertain and may follow different random distributions. The objective of these problems is to find an assignment of facilities to locations at each period that optimizes the material handling cost and the facilities relocation cost. This thesis has three contributions. First, it assesses the accuracy and efficiency of a *Parallel Shortest Path (PSP)* algorithm developed by Kolla (2015) to solve the DFLP. Second, it tests the efficiency on formulating a *linear network model (LNM)* for the DFLP and solving it with the network simplex algorithm implemented in AMPL, a commercial mathematical programming language, through numerical experimentation. Third, this thesis proposes a constrained shortest path network model to solve the SDFLP and experiments with small size instances. The SDFLP network model is an extension of the DFLP model in Balakrishnan et al. (1992).

1. INTRODUCTION

Tompkins et al. (2010) highlight the relevance of facilities planning from the point of view of size of the investment. They mention that U.S. businesses invested over a trillion dollars per year over the last five years in capital goods and from this amount, over 30% was spent on structures with a large part of this percentage spent on new constructions. In addition, Tompkins et al. (2010) state that over 8% of the United States gross national product has been annually spent on new facilities and that over $300 billion will be spent annually on facilities planning or re-planning.

Between 20%-50% of the total operating expenses in manufacturing in the US are attributed to material handling. Facilities planning deals with the reduction of this significant supply chain expenditure. Effective facilities planning can reduce material handling cost by 10-30% (Tomkins et al., 2010).

According to Ballou (2003), Logistics/Supply Chain embraces four major areas: customer service, inventory, location and transportation (see Figure 1). The facility layout problem (FLP) is a strategic planning problem that falls in the Location Strategy side of the triangle in Figure 1. The FLP occurs every time there is a need to plan for the arrangement of the facilities. The time horizons for the plan range typically from 3 to 10 years. Besides material handling cost, facilities relocation costs and improper location of facilities affect the performance of the supply chain. Reduction of material handling costs is crucial since it is incurred if routing all materials (raw materials, parts, sub-assemblies and assemblies) inside facilities or between them. A good location strategy has a positive impact on inventory levels, entire transportation costs, and customer service by reducing delays.

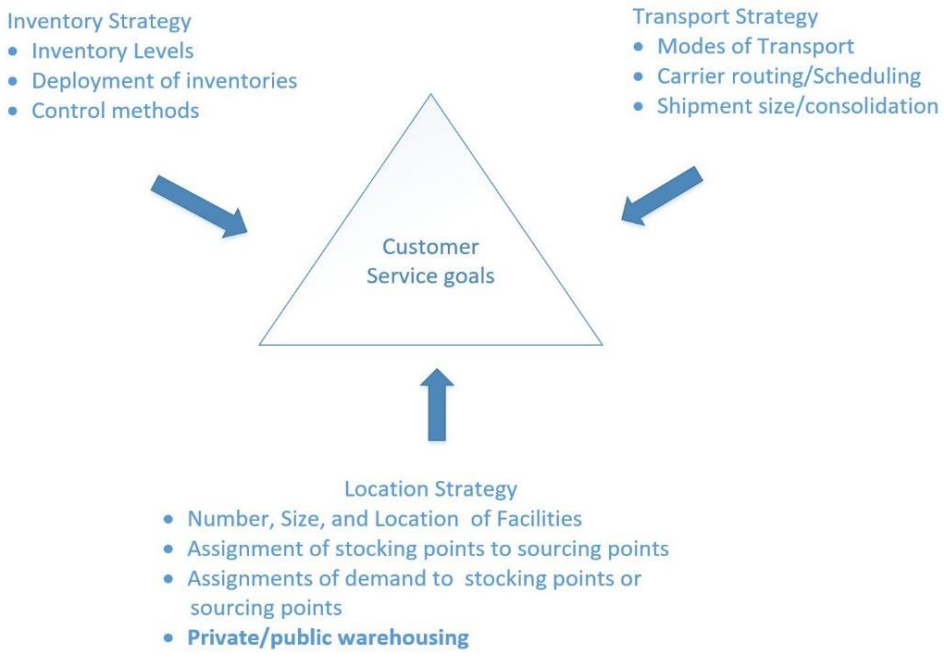


Figure 1. The logistics triangle (Ballou, 2003)

Ballou (1968) mentions that a physical distribution system can be conceptualized as several inventory storage points (nodal points) interconnected by a transportation network (links) (See Figure 2). Location and arrangement of inventories and warehouse facilities, transportation service choices, and inventory levels are major decision areas that concern managers designing distribution systems especially when demand and economic conditions change over time. This thesis researches on solution methodologies for solving two facility layout problems (FLP's): the Dynamic Facility Layout Problem (DFLP) and the Stochastic and Dynamic Facility Layout Problem (SDFLP).

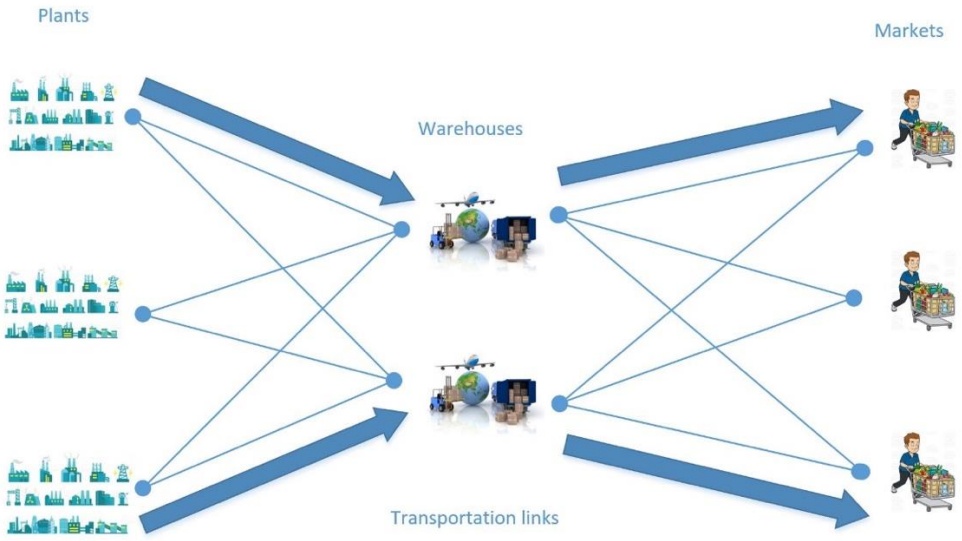


Figure 2. A representation of a segment of a physical distribution system (Ballou, 1968)

In the Dynamic Facility Layout Problem (DFLP), the flows of material/products between facilities (or departments) are known but vary over the time horizon. In the Stochastic Dynamic Facility Layout Problem (SDFLP), the flows of products between facilities are known only through probability distributions and vary over time. The objective of these problems is to find an assignment of facilities to physical locations for every period that optimizes the trade-off between material handling cost and facilities relocation cost. For these problems, the flows of material/products between facilities may vary because of fluctuations in product demand, introduction of new products, changes in product design, updates in production processes, new machinery, breakdowns and/or unexpected disruptions in the supply chain. The SDFLP is an extension to the DFLP and the Single-period or Static Facility Layout Problem (SFLP). The SFLP models as a Quadratic Assignment Problem (QAP) if the facilities are of equal size and the layout is divided into equal size locations. A practical example of facilities (machines or

departments) of equal size occurs in flexible manufacturing settings using multiple automated and multifunctional equipment such as Computer Numerical Control (CNC) machines (Moslemipour & Lee, 2012).

This thesis has three contributions. First, it assesses the accuracy and efficiency of an algorithm to solve the DFLP through numerical experimentation. The algorithm coded by Kolla (2015) is a variation of the Dijkstra's shortest path algorithm (1959), also presented by Tarjan (1983), and adapted to solve the DFLP. The instances selected for the computational study have 6, 10, 12, 15, and 30 departments and 5 years and they come from the Balakrishnan's repository (Balakrishnan & Cheng, 2000). The solutions from running Djikstra's implementation are compared to those published by Conway & Venkataramanan (1994), Balakrishnan & Cheng (2000), Baykosaglu & Gindy (2001), Balakrishnan et al. (2003), and McKendall & Shang (2006). Second, this thesis experiments with a linear programming model for the DFLP proposed in Balakrishnan et al. (1992) and assess its efficiency if solving it with the network simplex algorithm in AMPL, a commercial mathematical programming language, through extensive numerical experimentation. Third, this thesis devises a constrained network model to solve the SDFLP and solves this as a stochastic program. The SDFLP model extends the deterministic DFLP model in Balakrishnan et al. (1992). This works assesses the feasibility of solving the proposed model using a set of generated instances. This will be the first work in the literature modeling a constrained SDFLP under the network modeling approach.

The thesis consists of 7 chapters. Chapter 2 presents the mathematical models for the DFLP and SDFLP and a small example to understand the terminology and computations to evaluate the objective function for the DFLP. Chapter 3 presents a

literature review that summarizes the contributions under the solution approaches researched by previous authors. Chapter 4 presents the two solution methodologies studied in this thesis, the variation of the Dijkstra's Algorithm and the Network Simplex Algorithm. The examples presented on using the Network Simplex Algorithm illustrate at a small scale the solution steps performed by the AMPL software. Chapter 5 describes the experiments conducted to find the numerical results and an analysis of them. Chapter 6 applies the devised solution methods to solve a practical case study and Chapter 7 presents the conclusions of the thesis.

2. MATHEMATICAL MODELS FOR THE DFLP AND THE SDFLP

The Quadratic Assignment Problem (QAP) is a facility layout problem that models as a non-linear integer programming (NLP) problem with quadratic objective function. It aims to find an assignment of *n* facilities of about equal size to *n* layout locations that minimizes only the cost of material handling incurred in a single period.

2.1 Non-linear-programming formulation for the DFLP

The DFLP can be modeled as an extension to the QAP. The NLP formulation for the DFLP presented below is the one in Moslemipour & Lee (2012). Let *F* and *D* be two given matrices that can be asymmetric. $F = [f_{tkl}]$ contains the flows between any facilities or departments *k* and *l* at time *t* and $D = [d_{ij}]$ has the distances between any locations *i* and *j*. An optimal solution to the DFLP is to find the values for the decision variables, *x* and *y*, that minimize the cost of material handling (first term in equation 1) and facilities relocation cost (second term equation 1) and satisfy constraints (equations 2 - 5) over *T* periods of time. Table 1 summarizes the model notation.

$$\min z = \sum_{t=1}^{T}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{tkl} d_{ij} x_{tki} x_{tlj} + \sum_{t=2}^{T}\sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{tkij}\, y_{tkij} \qquad (1)$$

$$s.t \sum_{i=1}^{n} x_{tki} = 1, \qquad t = 1, \dots, T; k = 1,2, \dots, n \qquad (2)$$

$$s.t \sum_{k=1}^{n} x_{tki} = 1, \qquad t = 1, \dots, T; i = 1,2, \dots, n \qquad (3)$$

$$x_{tki} \quad \in \{0,1\} \qquad \forall t, k, i \qquad (4)$$

$$y_{tkij} = x_{(t-1)ki} * x_{tkj} \qquad \forall k, i, j, t \geq 2 \qquad (5)$$

Table 1. Model Notation for the NLP Formulation of the DFLP

Decision variables:	
x_{tki}	Binary decision variable that takes the value of one if department k is assigned to location i in period t and zero otherwise
y_{tkij}	Binary decision variable that takes the value of one if department k is shifted from location i in period $t-1$ to location j in period t and zero otherwise
Parameters:	
T	Length of the planning horizon, usually given in years
n	Total number of departments, which is also the total number of locations
f_{tkl}	Flows of material between facilities or departments k and l at time t
d_{ij}	Distances between any physical locations i and j
a_{tkij}	Cost of relocating department k from location i to j in period t

Constraints (2) and (3) are assignment constraints that guarantee every department is assigned to one location and every location has exactly one department. Constraints (3) and (4) state that the decision variables are binary. Thus, x_{tki} takes the value one if department k is assigned to location i in period t and zero otherwise. The decision variable y_{tkij} is one if the department k is shifted from location i in period $t-1$ to location j in period t and zero otherwise.

The parameter a_{tkij} representing the cost of relocating a single department can be modified to represent: (1) a fixed cost and independent of the department arranged or (2) a variable cost that depends on the department being moved but not on its locations. Rosenblatt (1986) and most other authors since then have considered a variable rearrangement costs (case 2). However, in many practical situations, the primary cost associated with the rearrangement of a facility is the fixed cost that results from the disruption, or possible shutdown, of its operations.

The right-hand side in Table 14.1 in Rardin (2017) lists a series of classical optimization problems that are believed not to be solvable in polynomial time, but this has not been proven. One of these problems is the QAP. The QAP and extensions of it such as the DFLP have been labeled as non-deterministic polynomial time hard (NP-hard) problems (Loiola, 2007). Rardin, 2017 states in its Principle 14.18 that if any single problem in NP-Complete or NP-Hard can be solved in polynomial time, then very member of NP is polynomially solvable, and P=NP. Consequently, unless P=NP, there can exist no polynomial time algorithm for any NP-Complete or NP-Hard Problem.

Because of the large size of the solution space for the DFLP formulated as a NLP (i.e. $n!^t$), exact solution approaches to DFLP's have been limited to instances with sizes about 6 departments and locations and 3 years. These works have used mostly Dynamic Programming (DP). Other solution approaches have solved the problem approximately using metaheuristics such as tabu search (TS), genetic algorithm (GA) and simulated annealing (SA).

The NLP model for the DFLP that was presented above can be extended to model the SDFLP where flows and relocation cost parameters are unknown with certainty and just the probability distribution for them is known. However, by introducing scenarios that represent forecasted values for these unknown parameters the NLP formulation gets easily intractable even for very small problem instances. This thesis researches in other methodologies different than NLP to solve the DFLP and SDFLP.

2.2 Graphical explanation of the DFLP

Figure 3 is a graphical example of a solution for a small size DFLP in which 3 departments had to be assigned to 3 locations in 3-time periods (i.e., years). The given

departments' relocation costs are presented in the small table on the top left side of the figure. *T* denotes the periods (i.e. years), MHC represents material handling cost, RC stands for the relocation cost to transition from one layout in a year to another layout in a consecutive year, and the layouts or permutations of departments are notated by the letter *Lxxx* where *xxx* is a permutation number.

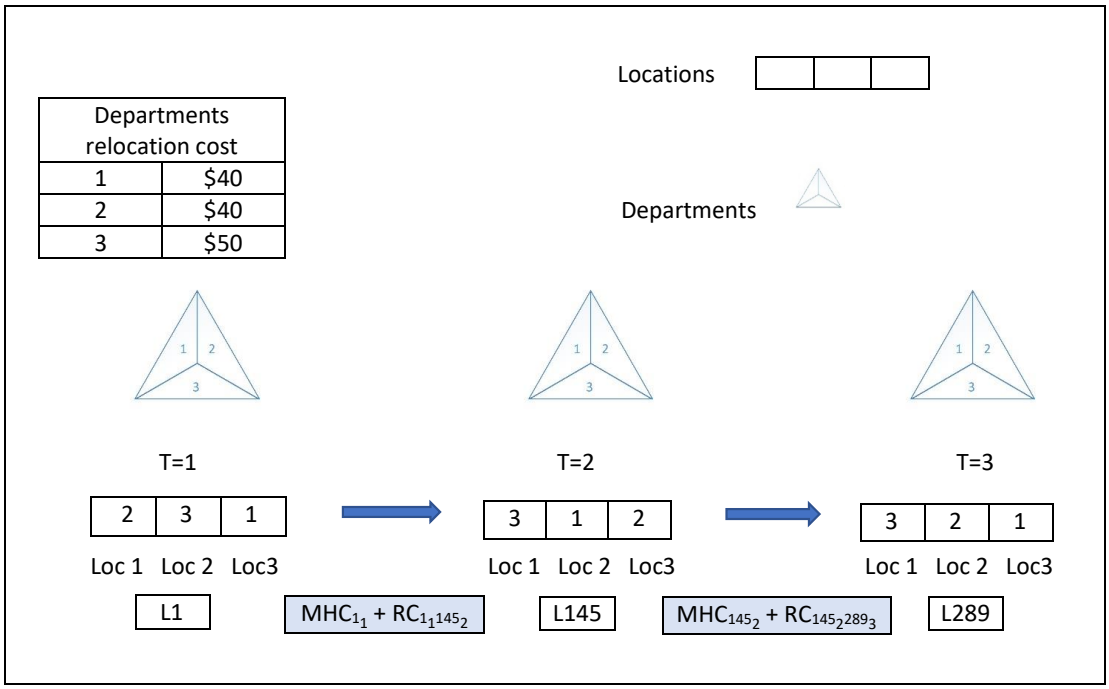


Figure 3. DFLP graphical example

The solution depicted in the Figure, indicates that in year *T=1* the departments 2, 3, and 1 are in locations 1, 2, and 3, respectively. In year *T=2*, department 1 is shifted to location 2, department 2 is shifted to location 3 and department 3 is shifted to location 1. Similarly, relocation of departments occurs if comparing layout *L145* in year two vs. layout *L289* in year 3. RC will be added to the MHC to compute the total cost of this solution to the DFLP.

The MHC computation for *L1* is calculated in Table 5. It results from the product of flows between departments, distances between locations and cost per unit of distance given the hypothetical layout, flow and distance matrixes presented in Table 2, 3, and 4.

Table 2. Hypothetical Layout or Permutation of Departments in Period 1

Department number	2	3	1
Location number	1	2	3

Table 3. Distances between Locations

Locations/Locations	1	2	3
1	0	3	2
2	2	0	7
3	5	6	0

Table 4. Flows between Departments

Departments/Departments	1	2	3
1	0	40	60
2	40	0	20
3	60	20	0

From Table 3, the distance between location 1 to 2 is 3 feet and from Table 4, the number of times product flows between departments 2 and 3 is 20. Thus, the flow-distance is 20*3 = 60. Similarly, the flow-distance computation is repeated for each pair of departments (taken from left to right and from right to left since the matrix of distance is non-symmetric). The MHC is the sum of all the flow-distance calculations (see Table 5) and it is equal to, 1160*0.25= $290 assuming the material handling cost is $0.25/feet.

Table 5. Example on Computing Material Handling Cost for a given Layout

Pairs of departments		Flows	Locations		Distance	Flows*Distance
2	3	20	1	2	3	60
2	1	40	1	3	2	80

Table 5. (Continued)						
3	1	60	2	3	7	420
1	3	60	3	2	6	360
1	2	40	3	1	5	200
3	2	20	2	1	2	40
				Total =		1160

From one period to another the departments change to different locations and a new MHC for the new layout is computed in a similar way as in table 5. The computed MHC of $290 for year 1 is added with the RC incurred to transition from period 1 to period 2 (from L1 to L145). If comparing *L1* from period 1 to *L145* in period 2, department 2 was moved from location 1 to location 3 and it costs of $50. Department's 3 and 1 were also moved and therefore RC is $130 (adding all numbers in the given departments' relocation cost table). Then the total cost incurred to go from *L1* in period one to *L145* in period two is $290 + $130=$420. The computation of MHC and RC is repeated for the lapse between years 2 and 3 using *L145* and *L289*. The total cost of the solution in Figure 3 will result from adding this cost to the $420 computed for the lapse between years one and two. In general, a department is shifted when due to the high flow between it and other(s) department(s) not currently close to it the MHC may be reduced given that the increase in RC's do not exceed those savings.

2.3 DFLP formulated as a network problem

Balakrishnan et al. (1992) indicated that the DFLP can be modeled as a linear network problem (LNP). The network proposed by Balakrishnan et al. (1992) to represent the DFLP as a LNP is in Figure 4 and its notation is in Table 6. Decisions occur at the end of the period. *L11* represents layout number *1* in the first period. *L22* represents a rearranged layout for the next period (layout 2 in period 2). Edges of the network connect

layouts from different periods and have costs, *C*, equal to the sum of the material handling cost of the layout in the previous period plus the relocation cost incurred from going from one layout to the next. The problem is to find the lowest cost dynamic layout plan as shown in the figure 4 by the red colored arcs. In Balakrishnan et al. (1992), the edges are also selected in such a way that the sum of the relocation cost incurred does not exceed a total amount of budget allocated to the rearrangements. The *linear network model (LNM)* and its notation, as in Balakrishnan et al. (1992), are presented below Table 6.

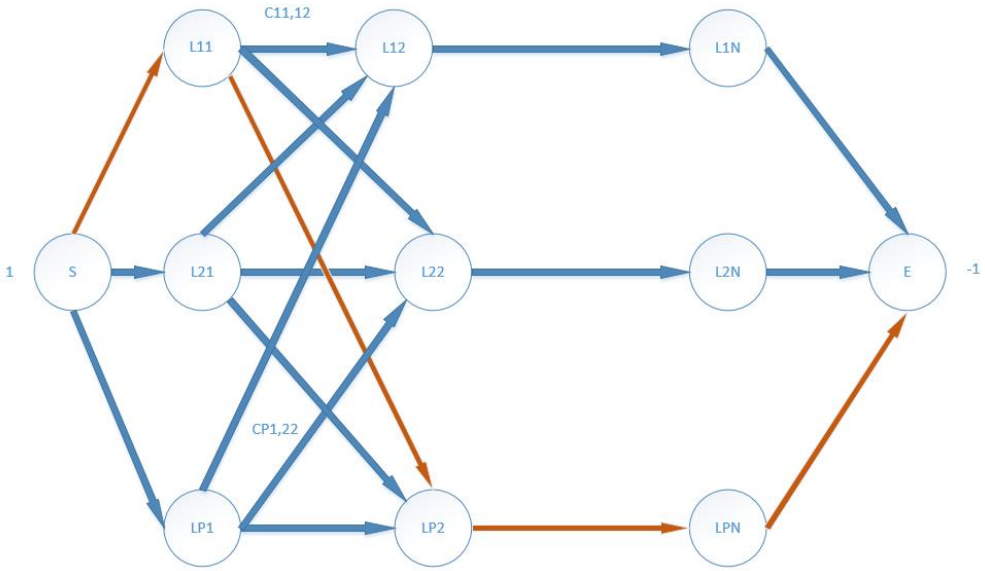


Figure 4. The constrained dynamic plant layout problem (Balakrishnan et al., 1992)

Table 6. Notation for the Network Problem in Figure 4

L_{it}	Static layout i in period t
S, *E*	Source and end nodes, respectively (dummy layouts)
P	Possible number of layouts in each period which is constant across periods

Table 6. (Continued)	
N	Number of time periods in the planning horizon

***Linear Network Model (LNM)* to solve the DFLP:**

$$Min\ z = \sum_{it} \sum_{k(t+1)} C_{it,k(t+1)}\, x_{it,k(t+1)} \qquad (6)$$

Subject to

$$\sum_{i1} x_{S,i1} = 1, \qquad (7)$$

$$\sum_{iN} x_{iN,E} = 1 \qquad (8)$$

$$\sum_{it} x_{it,k(t+1)} - \sum_{m(t+2)} x_{k(t+1),m(t+2)} = 0, \qquad for\ all\ k,t \qquad (9)$$

Model decision variables:

$x_{it,k(t+1)}$: The arc representing the sum of the MHC in layout i in period t and the *RC* incurred from layout *i* in period *t* to layout *k* in period *t+1*. The variable $x_{it,k(t+1)}$ is binary. It will be 1 if layout *i* is selected in period *t* and layout *k* is selected in period *(t+1)* and 0 otherwise.

$x_{S,i1}$: Arc connecting the source node S to layout i in period 1.

$x_{iN,E}$: Arc connecting layout i in period N to end node E.

Model parameters:

$C_{it,k(t+1)}$: The sum of the MHC in layout i in period t and the RC incurred from moving from layout *i* in period *t* to layout *k* in period *t+1*.

In the model, the objective function (6) minimizes sum of the RC and MHC. Constraint (7) guarantees that one unit is sent from the source node, *S* to one layout in period 1, constraint (8) assures that one unit arrives to the destination node, *E* and constraint (9) represents the flow conservation for all nodes. Note the difference in the meaning of the model decision variables for this model and the NLP presented in Section 2.1.

2.4 DFLP with a budget constraint modeled as a network problem

Balakrishnan et al. (1992) introduced the network model for the DFLP with the following additional budget constraint:

$$\sum_{it} \sum_{k(t+1)} RC_{it,k(t+1)} x_{it,k(t+1)} \leq B \qquad (10)$$

In this model, *B* is a parameter that represents the total available budget for relocations in the planning horizon.

2.5 SDFLP formulated as a network problem

This thesis proposes the following extension of the models in Sections 2.3 and 2.4 to solve a constrained SDFLP under a deterministic equivalent formulation (Birge, 2010) considering a total of *S*' scenarios and *N* time periods. The model is constrained because it considers limited budget. The three stochastic parameters in the model are the budget available for relocations (A), the cost of relocating the departments (and thus RC is stochastic), and the matrices containing the flow of material between departments (and thus *C* is stochastic since *C* is the sum of MHC and RC). The model and its notation are presented below. Given that this model is proposed by the author of this thesis the notation has been slightly changed vs. the one in Balakrishnan et al. (1992). To have a clearer notation, indexes *i* and *k* that represent a layout number and index s that represents a

scenario are notated as subscripts and the time indexes (i.e. *t*, *t+1*, *N-1*) are notated as sub-subscripts.

$$min\ z_{stoc} = \sum_{s=1}^{S'} \sum_{i_t}^{L_t} \sum_{K_{t+1}}^{L_{t+1}} p_s C_{si_tK_{t+1}} x_{i_tK_{t+1}} - \sum_{s=1}^{S'} p_s y_{s_{(N-1)}} (1+r)^{-(N-1)} \quad (11)$$

$$\sum_{i_1} x_{Start\ i_1=1} \quad (12)$$

$$\sum_{i_N} x_{i_N\ E=1} \quad (13)$$

$$\sum_{i_t} x_{i_t\ K_{i+1}} - \sum_{m_{t+2}} x_{K_{t+1}m_{t+2}} = 0 \qquad for\ all\ k,t \quad (14)$$

$$\sum_{i_t}^{L_t} \sum_{k_{t+1}}^{L_{t+1}} RC_{s_{i_t}K_{t+1}} x_{i_tK_{t+1}} + y_{s_t} = B_{s_t} \qquad \forall s, t = 1, \dots N-1 \quad (15)$$

$$B_{S_1} = A_{s_1} \qquad \forall s \quad (16)$$

$$B_{S_{t+1}} = A_{s_{t+1}} + y_{s_t} * (1+r) \qquad \forall s, t = 1, \dots, N-2 \quad (17)$$

$$B_{S_t} \geq 0 \qquad \forall s, t = 1,\dots,N-1 \quad (18)$$

$$y_{s_t}\ unrestricted\ in\ sign\ (urs) \qquad \forall s, t = 1, \dots, N-1 \quad (19)$$

Model decision variables:

$x_{i_tk_{t+1}}$: The arc representing the sum of MHC in layout *i* in period *t* and the RC incurred if changing from layout *i* in period *t* to layout k in period *t+1*. The variable $x_{i_t,k_{t+1}}$ is binary. It will be 1 if layout *i* is selected in period *t* and layout *k* is selected in period *(t+1)* and 0 otherwise.

$x_{Start\ i_1}$: Arc connecting source or start node (Start) to layout i in period 1.

$x_{i_N,E}$: Arc connecting layout i in period N to end node E.

y_{s_t}: If positive, it represents the amount of budget not used for relocations occurring between year t and t+1 under scenario s. If negative, it represents the extra money to borrow to do relocations occurring between period *t* to *t+1* under scenario *s.*

B_{s_t} : Total available money for relocations between year t and *t+1* under scenario *s*.

Model parameters:

$C_{si_tk_{t+1}}$: The sum of the MHC in scenario *s* for layout *i* in period *t* and the cost of rearranging layout *i* in period *t* to layout *k* in period *t+1*.

p_S : Probability of scenario *s*.

$RC_{s_{i_t K_{t+1}}}$: Cost of rearranging from layout *i* in period *t* to layout *k* in period *t+1* under scenario s.

A_{S_t}: Available or allocated budget for the relocations occurring between period t and t+1 under scenario s

r: Annual interest rate

In the SDFLP model above, equation (11) minimizes the cost of material handling and relocations and considers the earnings or loses resulting from money not used for relocations or borrowed for doing those relocations assuming an interest rate *r*. Constraints (12) to (14) are the same as for the DFLP model with budget constraint proposed by Balakrishnan et al. (1992). Constraints (15) to (17) correspond to flow conservation constraints. Constraint (15) says that for a given period, the amount used in relocations plus any money borrowed or left (decision variable *y* can be negative or positive as explained in the decision variables list) should be equal to the total available money for relocations in such period and scenario. Constraint (17) indicates that the total available money for relocations in a period other than the first one is equal to the available or allocated budget

for relocations in such period plus money (considering interest) that could have been left from the previous period or that must be repaid. Constraint (16) is a border constraint and indicates that the total available money for relocations in period 1 is equal to the available or allocated budget since there is no money left from a previous period to be brought to the period 1. Thus, the model assumes $Y_{s_0} = 0$.

3. LITERATURE REVIEW

This chapter presents relevant work on facility layout problems (FLP's) besides the work in Balakrishnan et al. (1992) mentioned in the previous chapter. The chapter reviews: (a) seminal works on solving FLP's, (b) approaches to solve DFLP and SDFLP's, (c) and a problem arising in supply chain known as the dynamic warehouse location problem (DWLP) which is closely related to the DFLP. The focus of this review is on problems that model as extensions of the Quadratic Assignment Problem (QAP) and consequently assume that the facilities are about the same size.

3.1 Surveys on recent advancements on solving facility layout problems (FLP)

Balakrishnan & Cheng (1998) mainly discuss approaches to solve DFLP's with equal sized departments. The approaches are Dynamic Programming, Computerized Relative Allocation of Facilities (CRAFT) heuristic, Genetic Algorithms, Tabu Search, Cutting Planes, Branch and Bound and Cut Trees. The authors also review contributions that deal with stochastic material flow in DFLP. The last section of their paper reviews the DFLP with unequal size departments.

Kulturel-Konak (2007) emphasizes on the uncertainty surrounding production environments and on the relevance of designing robust and flexible facilities. Her work reviews previous work on DFLP and SDFLP's and mentions two types of uncertainties: (i) internal disturbances, such as equipment breakdowns, variable task times, queuing delays, rejects and reworks and (ii) external forces, such as uncertainties in the level of demand, product prices, and product mix. She discusses exact, heuristics, meta-heuristics and hybrid solution methods and indicates that the future research direction is moving to address the uncertainties and changing scenarios in the production environment.

Moslemipour, Lee, & Rilling (2012) focus on discussing dynamic and robust layouts, mathematical models and solution approaches for the FLP. The mathematical models presented are Quadratic Assignment Problem (QAP), Quadratic Set Covering Problem (QSP), Mixed Integer Programming (MIP) and Graph Theoretic Models (GT). The reviewed intelligent solution approaches are Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO), Greedy Randomized Adaptive Search Procedure (GRASP), Particle Swarm Optimization (PSO), Artificial Immune System (AIS), Expert System (ES), Fuzzy System (FS), Artificial Neural Networks (ANN). The advantages and disadvantages of these solution approaches are summarized in detail.

3.2 Heuristics derived from exact methods to solve static or single-period Quadratic Assignment Problems (QAP)

Burkard & Bonniger (1983) propose a cutting plane algorithm without Bender's decomposition as an update to the methods proposed by Balas and Mazzola (1980). The authors differentiate their cutting plane method (CPM) by proposing another linearization technique. They further improve the solution given by the CPM by doing pairwise exchanges. In this thesis, the approach to solve the DFLP is through a *Linear Network Model (LNM)* instead of using the NLP formulation used by these authors.

3.3 Dynamic Programming (DP) to solve the DFLP

This section presents different previous works using DP to solve the DFLP. The DP approach has high resemblance with the approaches used in this thesis. As stated in the introduction, this thesis solves the DFLP with a variation of the Dijkstra algorithm and compare such solution approach to the one using a *LNM* (section 2.3). Rosenblatt, (1986)

presents a formulation for the static facility layout problem (SFLP) as a non-linear integer program (NLIP). In the formulation, the number of locations is equal to the number of departments. The author mentions that problems where the number of locations is different than the number of departments can be solved with this model by introducing dummy departments or locations. He obtains an exact solution for a six departments DFLP using exact DP and derives lower and upper bounds on the optimal solution. For DFLP's with more than 6 departments and 3 periods, Rosenblatt, (1986) suggest a heuristic that selects random layouts from solving the SFLP using CRAFT. The layouts are the input for each stage of the DFLP algorithm.

Lacksonen & Enscore (1993) benchmark the exchange algorithm, cutting planes, branch and bound, DP, and cut trees to solve a quadratic assignment formulation of the DFLP. The authors generate the instances to test the algorithms. The instances have 6 to 30 departments and 3 to 5 periods. The cutting plane algorithm provided the best solution with a reasonable computational time.

Urban (1998) applies the incomplete DP concept to solve the DFLP with fixed rearrangement costs incurring in low computational times. For a T-period DFLP, the incomplete method solves only $T(T+1)/2$ static facility layout problems (SFLP's) that correspond to the cases where relocation cost is incurred. Using these layouts, a solution is found using DP. The author also solves the DFLP using GRASP and Initialized Multi Greedy Algorithms and compares those results to the ones from the incomplete DP method. The work presents also an improved lower and upper bound to the DFLP.

3.4 Heuristic and meta-heuristic approaches to solve the DFLP

This section presents seminal contributions regarding heuristics methods to solve the DFLP. In the Numerical Results chapter, the methodologies proposed in this thesis will be compared to results from some of authors mentioned in this section. Conway & Venkataramanan (1994) are the first authors using GA to solve the DFLP with a budget constraint. GA is a meta-heuristic that applies concepts and terminology brought from genetics to find a near optimal solution. GA most relevant elements are generation, chromosomes, population, survival of the fitness, crossover, and mutation. The GA proposed is tested in an $n=6$ departments (and locations) and $t=5$ periods problem. The chromosome size is n×t, or 30 digits, representing the location of every department in each period. Also, for each period there are $n!=720$ chromosome strings in the population. Based on the fitness function (i.e. the cost function), two strong chromosomes are selected, split and swapped. It may end up in infeasible solutions like repeating a department in one period. Replacement of the invalid chromosome digits produces feasible solutions. The GA repeats the process of selecting pairs of chromosomes, splitting and swapping to produce new feasible solutions at each iteration of the algorithm.

Kaku & Mazzola (1997) use Tabu Search (TS), a technique that does pairwise interchanges between departments in the local neighborhood of a solution and keeps a tabu list to avoid cycling. A diversification strategy is used to ensure that different regions of the search space are explored. Their TS heuristic is a two-stage procedure. In the first stage, diversification allows to get several different solutions. The best solutions generated are fed into the second stage for intensification of the search and determine a final solution.

Problems from Lacksonen & Enscore (1993) repository with up to 30 departments and 5 periods took 3 hours, on average, to run on a Pentium 200 MHz PC.

Balakrishnan & Cheng (2000) propose a GA method that outperforms the one in Conway & Venkataraman (1994). Later, Balakrishnan et al. (2003) and Balakrishnan & Cheng (2006) produce a hybrid algorithm that combines GA and DP. DP is used at the crossover step to find the best multi-period layout based in fitness. This hybrid algorithm takes advantage of DP and GA without being computationally prohibitive.

Baykasoglu & Gindy (2001) propose a SA algorithm with a simple, but effective, data structure and neighborhood generation mechanism. Solutions are in a two-dimensional matrix (periods in the rows, locations in the columns, and departments are entries of the matrix). Neighborhood solutions result from swapping elements in the rows of the matrix. The algorithm is applied to small test problems from the literature. It finds the optimal solutions and performs better than the DP from Rosenblatt, (1986) and the GA of Conway & Venkataraman, (1994). Additional computational experimentation is done with the data set from Balakrishnan & Cheng, (2000) which contains problems for 6, 15, 30 departments for 5 and 10 periods. Comparisons made with the GA in Balakrishnan & Cheng, (2000) show that the proposed SA performs considerable well. However, computational time increases considerably as problem size gets bigger.

McKendall & Shang (2006) are the first ones to apply Hybrid Ant Systems (HAS) to the DFLP and develop three different versions of the heuristic (HAS I, HAS II, and HAS III). Results show that the proposed versions perform well on Lacksonen & Enscore (1993) and Balakrishnan & Cheng, (2000) data sets. A pairwise exchange technique is used in HAS I to improve the initial and updated solutions from using the pheromone trail matrix.

HAS II uses SA as the local search heuristic and the ideas in HAS I. HAS III adds a look-ahead/look-back strategy to the pairwise exchange heuristic within the HAS I heuristic. HAS I, II, III obtained the best solutions for 19, 30, and 36 problems from Balakrishman & Cheng (2000) repository, respectively.

Baykasoglu et al. (2006) propose an ant colony heuristic for solving complex combinatorial optimization problems like DFLP. The authors came out with competitive solutions however, Mckendall & Shang (2006) gave a few better results.

Sahin et al. (2010) propose a simulated annealing algorithm to solve the DFLP with a constraint on the budget available for relocations. They compare the proposed heuristic with the one in Baykasoglu et al. (2006) because they state that Baykasoglu et al. (2006) and Balakrishnan et al. (1992) are the only two previous studying the DFLP with budget constraints. The problems compared have 5, 15 and 30 departments and 5 and 10 periods. The average improvements obtained by Sahin et al (2010). vs. Baykasoglu et al. (2006) are 1.27& for the instances of size 5, 6.19% for the instances of size 15 and 5.59% for the instances of size 30.

3.5 Simulation approaches to solve the DFLP

A novel perspective to solve DFLP's is introduced by Azimi & Charmchi (2012). They proposed a heuristic algorithm to solve a DFLP with budget constraint combining linear/integer programming and simulation. After solving a linear relaxation of the non-linear model for the DFLP, the decision variable is interpreted as the probability of assigning a department to a location. In the experiments, the budget is split equally for each period and for some problems the budget was set equal to the rearrangement costs. This work proved to be efficient in computational time and was tested in problems of up to 30

departments. They are the second authors after Sahin et al. (2010) to research on a DFLP model with two constraints that limit the expenditure on relocations and tie the leftover budget from a previous period to the total available budget in a current period.

3.6 Stochastic and Dynamic Facility Layout Problems (SDFLP)

Palekar et al. (1992) use exact methods based on DP and heuristics to solve for the first time an SDFLP under fixed and rolling horizons options. The work provides lower and upper bounds to the problem optimal solution. In the exact approach, this work exemplifies the combination of the use of integer programming (IP) and DP. The author found exact solutions for problems with up to 12 departments and 8 periods. Also, the approximate methods tested successful for problems with up to 40 departments and 8 periods. Using a rolling horizon produced good results if compared to fixed horizon. The main contribution of this work is a drastic reduction on computational time since heuristics reduced the number of layout combinations explored in the DP procedure.

Benjaafar & Sheikhzadeh (1997) find the most flexible layout over a set of demand scenarios. In addition to variability of product mix and product demand, the authors allow duplication of the same department type within the facility. In fact, disaggregation and distribution of a department throughout the facility is not a new idea. Earlier, Montreuil et al. (1993) introduced the concept of holographic layouts for systems operated in high volatile environments. A holographic layout allows the spreading of machines in a facility. The authors assume that: (1) each sub department may consist of more than one machine, (2) all sub departments of the same type may not necessarily have the same capacities and (3) material flow between departments is a decision variable. The material handling cost is reduced, and the solution adjusts to the fluctuations in flow patterns and volumes. The

authors use a heuristic method to solve the problem. It is an extension to the CRAFT algorithm where a flow volume allocation problem is solved first in an exact way. They show that duplicates of the same departments can significantly reduce material handling cost while effectively coping with fluctuations in flow patterns and volumes. However, most of the cost reduction occurs with relatively few duplicates.

Krishnan et al. (2008) work on minimizing uncertainties from multiple demand scenarios in single and multi-period facility layout problems. The problems are solved under minmax approach and minimize the total expected loss (MTEL) approaches. Under the minmax approach, the objective is to minimize the maximum loss considering all possible scenarios. The common layout to be used under all scenarios is found using GA. The results show that these models are effective in reducing the risks that are associated with facility layout design under uncertain environments.

Tayal et al. (2016) propose a methodology to solve a sustainable stochastic demand flow facility layout problem. They integrate meta-heuristic techniques such as SA, chaotic simulated annealing (CSA) and hybrid firefly algorithm (FA/CSA) to generate the layouts. The best layouts among the generated layouts are filtered using data envelopment analysis (DEA) and applying multiple attribute decision making (MADM) approaches such as TOPSIS, IRP and AHP in association with aggregate ranking methods and integer linear programming (ILP).

Vitayasak et al. (2017) solve SDFLP's assuming product demands follow exponential, normal, and uniform distributions and using GA and the Backtracking Search algorithm (BSA) as solution methods. The novel modified BSA consists of five processes: initialization, selection-I, mutation, crossover and selection-II. The computational

experiments compared five algorithms GA, BSA and modified BSA's (mBSA1, mBSA2, mBSA3). Results show that GA performs much better than BSA in terms of minimizing the cost. However, BSA took 55% less time than GA and thus the authors suggest that BSA is suitable for large, computationally intensive optimization problems.

Tayal et al. (2018) propose Simulated Annealing (SA) and Chaotic Simulated Annealing (CSA) meta-heuristics to solve a Multi Objective Stochastic Dynamic Facility Layout Problem (MO-SDFLP) to solve the location-based demand problem on the facility during disasters. The main aim is to find a layout that responds to the sudden demand variations occurring in disaster relief situations faced by supply chains. The two considered objectives are to minimize the flow times distance while maximizing the department's closeness or adjacency desirability, which is in many cases subjective information related to noise, heat, dust, flow of material, etc. Tayal et al. (2018) test their proposed methods on the Moslemipour & Lee (2012) problem with 12 Departments and 5 periods and Gaussian (i.e. normal) distribution product demand. For further testing the capabilities of MO-SDFLP on a high-demand disaster situation the SA and CSA algorithms are used to solve a problem with 30 Departments and 5 periods. The results showed that CSA performed better than SA.

3.7 Dealing with planning horizon and other practical considerations that should be included when solving the DFLP

Azadivar & Wang (2000) solve a single-period FLP by considering other dynamic characteristics and operational constraints such as the time involved in moving the material and number of transporters. The objective function is not to minimize material handling cost but to minimize average cycle time. GA is used to optimize the facility layout for cycle

time and productivity. Simulation is used to evaluate the performance of the system. The method proved very effective to find near-optimal solutions. However, the time consumed by the computer simulation is significant.

Balakrishnan & Cheng (2009) compare the Urban heuristic (1998) and approximate DP to solve a DFLP starting from layouts generated with the CRAFT heuristic and randomly. The comparisons were done under fixed vs. rolling horizons options. The authors mention that algorithms developed for the fixed horizon case are not as effective as those developed for rolling horizons. They also indicate that it is hard to pick up an algorithm that runs effectively under both fixed and rolling horizons. The effectiveness of the rolling plan horizon algorithm is compared under various scenarios. The authors conclude that further research is needed to coin an effective self-adjusting algorithm under rolling horizons.

Appendix A presents a table that summarizes the DFLP works reviewed in this Sections 3.1-3.5 and 3.7. Appendix B presents a table that summarizes the SDFLP works reviewed in Section 3.6. The current gaps in the literature are to: (1) find more efficient and accurate ways to solve DFLP for instances with more than 10 departments (10 locations) with 3-5 periods and (2) provide a stochastic model and a solution approach for the SDFLP that does not rely on DP or meta-heuristics. This thesis attempts to fill this gap. In the last part of this section, we briefly review the Dynamic Warehouse Location Problem (DWLP), a supply chain problem that closely relates to the DFLP.

3.8 Dynamic Warehouse Location Problem (DWLP)

Ballou (1968) mentions that in a rapidly changing economy, warehouse location-relocation is also a dynamic decision problem, yet most existing location models used to

solve the problem are static. Ballou applies the DP technique to find a location-relocation plan for a single warehouse that will maximize total profit in a given planning period of five years. This DWLP involves the tradeoff between expected profit and time to relocate warehouses. Relocation profit depends on decision makers periodical review of new input data about the location of the warehouse. When future demand and economic data can be forecasted only for a short time horizon and the forecasted demand is high, it is not a good idea to relocate since at the time of relocation the demand might have changed, and the company may end up with a profit much lower than predicted. Accurate long-term forecasts provide the best scenario for the decision makers because it gives them enough time to relocate warehouses. As relocation is also concerned with land purchase, construction, lease negotiation, financing, closing and starting operations, accurate long-term forecasts put relocation as risk free, because of the ample time, and low chances of changes for demand and economic data after relocating.

Sweeney and Tatham's (1976) propose an approximate method to solve a DWLP that combines DP and Mixed Integer Linear Programming. They opt for a computationally efficient DP algorithm that consists of using as states in each period only the R best static configurations for the warehouse on each period. The generation of the best configurations involves solving the MILP multiple times by adding constraints that avoid repetition of the previous static solution. This approach was illustrated in a problem with five-year planning horizon, two plants, five warehouse locations, and 15 customer zones.

DWLP could be extended to include the identification of the best layout for products inside the warehouse. High demand of a product makes a product of high priority. The warehouse will save in relocation and material handling costs by finding the optimal

location for the products. Furthermore, products can be grouped by using inventory stratification methods, such as ABC and weighted linear optimization. Such extended DWLP could include multiple objectives such as maximize the overall profits in the warehouse location decision and achieve top-notch customer service over the time horizon.

4. METHODOLOGY

This chapter consist of 4 sections. Section 4.1 explains a variation of the Dijkstra's algorithm used in this thesis to solve the DFLP modeled as a network and presented in Chapter 2 (Section 2.3). Section 4.2 briefly presents Dynamic Programming (DP), a closely related methodology. Section 4.3 explains and exemplifies the simplex for networks algorithm (SNA), the second methodology used in this thesis. Section 4.4 briefly discusses how these methodologies were implemented for developing the computational study.

4.1 Parallel Shortest Path (*PSP*) algorithm for solving the DFLP

This thesis experiments with the parallel implementation to solve the DFLP as a shortest path problem developed by Kolla (2015). This implementation is notated as *Parallel Shortest Path (PSP).* It is a slight variation of the Dijkstra's algorithm (Dijkstra, 1959), also presented in Tarjan (1983) and Rardin (2017). Following is a presentation of the variation of the Dijkstra's algorithm implemented by Kolla (2015) to solve the DFLP.

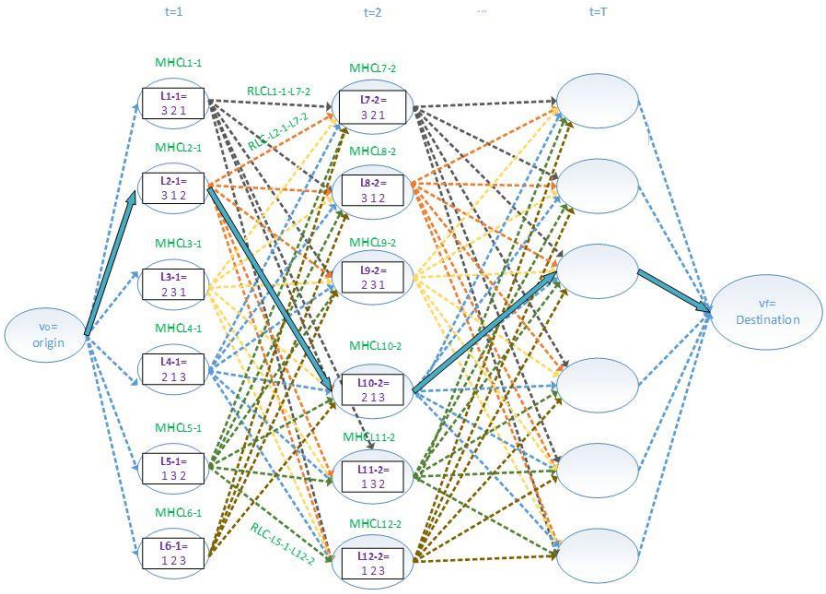


Figure 5. A network representation of the DFLP

Figure 5 presents the DFLP modeled as a network problem in a directed weighted graph *G*. To make Figure 5 simple, the DFLP presented has three facilities, three locations and *T* time periods. In general, G has *T*n! + 2* vertexes or nodes where *T* is the total number of periods and *n!* is the total number of layouts considered in each period if the problem has *n* departments and *n* locations. G has (T-1)*(n!*n!) + 2n! arcs. Two dummy nodes v_o and v_f are added to the network and they represent the single origin and the destination.

Node *v* in the graph define a layout of facilities (i.e. assignment of facilities to locations), represented as a permutation π of integers, *{1 … n}*. Material handling cost *(MHC)* for node *v* is computed using the permutation π, the flow (*F={ftkl}*) and distance (*D={dij}*) matrices as exemplified in Section 2.2. *MHC* is the sum of all feasible products of flow between departments and distances between locations. There is a unidirectional edge e_{rs} from node v_r to v_s if v_r and v_s are in two different periods (i.e. *r* is in *t-1* and *s is in t*, this means v_r precedes v_s). The *weight* $W(e_{rs})$ on each edge can be computed as $W(e_{rs}) = RCe_{rs} + MHCv_s$ where *RC* represents the relocation cost. The value for RCe_{rs} results from comparing the permutations in v_r and v_s and adding the relocation costs for all departments that have shifted their locations from one period to the next one.

The *length* of a path in *G* is the sum of its edge *weights*. The *cost* to travel from a node v_r to a node v_s is the *minimum length* of a path from v_r to v_s. However, note that in the DFLP, the only interest is to find the *shortest path* or *minimum-cost path* from v_o to v_f. The *shortest path DFLP* algorithm uses a tentative cost function *C[v]* such that when the algorithm ends $C[v_f]$ has the total cost to travel (i.e. to go) from v_o to v_f. The steps in the variation of the Dijkstra's algorithm implemented by Kolla (2015) to solve the DFLP are listed below.

Variation of the Dijkstra's algorithm to solve the DFLP:

Initialization Step

- *E*ach vertex or node is in one of three states: unlabeled (and with infinite cost), labeled (i.e. temporary labeled) or scanned (i.e. permanently labeled).
- At the beginning v_o is the only vertex labeled and $C[v_o]= 0$. The vertexes v_s that connect directly to node v_o are temporarily labeled with a cost $C[v_s]$ equal to the sum of $RC_{os} + MHCv_s$ which is equal to $MHCv_s$ since RC_{os} is zero. All other nodes are unlabeled and have infinite cost. The index i is set equal to 1 and the scan step is performed.

Scan or Processing Step

- Select vertex v_i and change it from labeled to scanned. For each edge e_{rs} such that $C_{v_i} + RC_{v_i v_s} + MHC_{v_s} < C_{v_s}$ replace C_{v_s} by $C_{v_i} + RC_{v_i v_s} + MHC_{v_s}$ and make v_s labeled. If $C[v_s]$ was replaced, set the predecessor node of s, $p[s]$, as vertex. Increase i to $i+1$ and repeat the scan step until no temporary nodes remain.

Comments about the algorithm

Storing the predecessor node information permits to find the shortest path from v_o to v_f when the algorithm ends. This is done by going backwards from v_f to v_0 and using the information in $p[s]$. There are a couple of differences to highlight between this variant of the Dijkstra's algorithm to solve the *DFLP* and the algorithms originally proposed by Tarjan (1983) and Dijkstra's (1959). In the *PSP* implementation, finding the *shortest distance* corresponds to finding one layout for each period that minimizes the *total cost* in equation (1) of the model presented in section 2.1 or equivalently the total cost in equation (6) of the model presented in 2.3. Besides, in each iteration, the selection of the node to

label as scanned (Tarjan, 1983) or permanent (Taha, 2013) among the temporary nodes does not require to loop over the temporary labeled nodes to find the one of minimum *cost (i.e. distance).* It is because there is no interest in identifying the nodes that are closer to the source node v_o in increasing order of *cost.* It reduces the computational time of the *PSP* implementation. Besides, the DFLP network has no edges between nodes that belong to the same period or are separated by more than one period. It also simplifies the number of comparisons to perform by the *PSP* implementation.

In the DFLP the problem size rapidly grows for instances with more than $n = 6$ departments and $t = 3$ periods. To cope with this issue, the *PSP* implementation generates a large number N of layouts for each period t where $(N<n!)$ and proceeds to execute the shortest path algorithm previously described. Since the problem exemplified in figure 5 is small (3 departments, 3 locations, T periods), all $n!$ layouts are in the network. Consequently, the reader sees a repetition in the layouts (or permutations) in the nodes over the periods.

Regarding the parallelization implemented by Kolla (2015), it focused on computing the MHC of the nodes in a parallel fashion using two reduction operations available in OpenMP. Kolla also observed that the computation of MHC and RC are independent and can be executed in parallel. However, the time to compute MHC is significantly larger than the time to compute RC. Then, there was no great reduction in computational times when implementing the last parallelization idea.

4.2. Solving DFLP using Dynamic Programming (DP)

Alternatively, the *shortest path* problem can be modeled and solved with the Dynamic Programming (DP) methodology. However, this approach still will have the curse

of dimensionality mentioned in the previous section. The main idea in a DP approach is to decompose the problem into *sub-problems* (Taha, 2013). Computations are performed recursively in such a way that the optimum solution of one *stage* (i.e., a *sub-problem*) is used as an input to the next *stage*. The optimum solution is obtained when the last *stage* is reached. In the DFLP a *stage* represents a period. Using the DFLP notation introduced in the previous paragraphs, the forward recursive equation to solve the DFLP under a DP approach is given by equation (6).

$$C_{v_s}^{t} = \min_{v_r \in (stage\ t-1)} \left\{ C_{v_r}^{t-1} + RC_{v_r v_s} \right\} + MHC_{v_s}^{t} \qquad (6)$$

The recursive equation (6) express the shortest cost to *state* or node v_s in the next *stage t*. It links successive *stages* in a way that permits optimal decision for every *state* or node v_s in a future *stage t* independently of the decisions already made in all preceding stages. DP is an operations research approach appropriate for exactly solving multi-period or multi-stage problems.

4.3 Network simplex algorithm for solving DFLP modeled as a network problem

The network simplex method (NSM) is an adaptation of the bounded variable primal simplex algorithm. It can be used to solve Minimum-Cost Network Flow Problems MCNFP such as shortest path, maximum flow, transportation, assignment, transshipment and critical path problem. The primal and the dual for the MCNFP are:

Primal:

$$\boldsymbol{min\ C^T X}$$

$$\boldsymbol{s.t., Ax = b}$$

$$\boldsymbol{x \geq 0}$$

Dual:

$$\boldsymbol{Max\ b^T W}$$

$$\boldsymbol{s.t.,\ W^T A \leq\ C^T}$$

$$\boldsymbol{W\ unrestricted\ in\ sign\ (URS)}$$

The MCNFP variables are the arcs. Iterations are done until the optimal solution is achieved, this means when no non-basic variable (i.e., arc) is eligible to enter to the basis. The first iteration starts by arbitrarily selecting a spanning tree. Second iteration and the rest of the iterations identify the non-basic arc to enter. The arc with the maximum violation of the optimality condition causes the maximum decrease in the objective function per unit change in the value of flow on the selected arc. It will enter to the basis in the next iteration and a new objective function cost will be computed. The detailed steps for the NSM as discussed in Winston (2004) are presented below.

4.3.1 Network simplex method (NSM) procedure

Step 1: Determine the initial basic feasible solution (bfs) by selecting a spanning tree arbitrarily. Indicate non-basic variables at their upper bound by dashed arcs.

Step 2: Compute $W_1, W_2, \ldots W_n$ (simplex multipliers or dual values) by solving $W_i - W_j = C_{ij}$ for all basic variables X_{ij} and giving the value W= 0 to one of the W_i's. For all non-basic variables, determine the reduced cost coefficient $\bar{C}_{ij}$ from $\bar{C}_{ij} = W_i - W_j - C_{ij}$. The current bfs is optimal if $\bar{C}_{ij} \leq 0$ for all $X_{ij} = L_{ij}$ and $\bar{C}_{ij} \geq 0$ for all $X_{ij} = U_{ij}$. If the bfs is not optimal, choose the non-basic variable that most violates the optimality conditions as the entering basic variable.

Step 3: Identify the cycle (there will be exactly one!) created by adding the arc corresponding to the entering variable to the current spanning tree of the current bfs. Use

conservation of flow to determine the new values of the variables in the cycle. The basic variable that exits the basis will be the variable that first hits its upper or lower bound as the value of the entering non-basic variable is changed.

Step 4: Find the new bfs by changing the flows of the arcs in the cycle found in step 3. Go to step 2.

Following are two examples on the network simplex method. Example 1 show some of the steps for solving small size Shortest Path Problem. Example 2 show the steps for solving a Transshipment Problem without bounds on the variables.

4.3.2 Example 1: solving a Shortest Path Problem (SPP) using the NSM

Figure 6 depicts a Shortest Path Problem. In this network, the costs are given in $. As a first step, one feasible solution (i.e. arbitrarily chosen spanning tree with n-1 arcs) is identified as shown Figure 7. A spanning tree is a tree that connects all nodes and does not form a cycle. All the five arcs represent basic variables (X_{13} and X_{56} are at its lower bound and the others at its upper bound). The omitted arcs (X_{25} and X_{35}) represent non-basic variables.

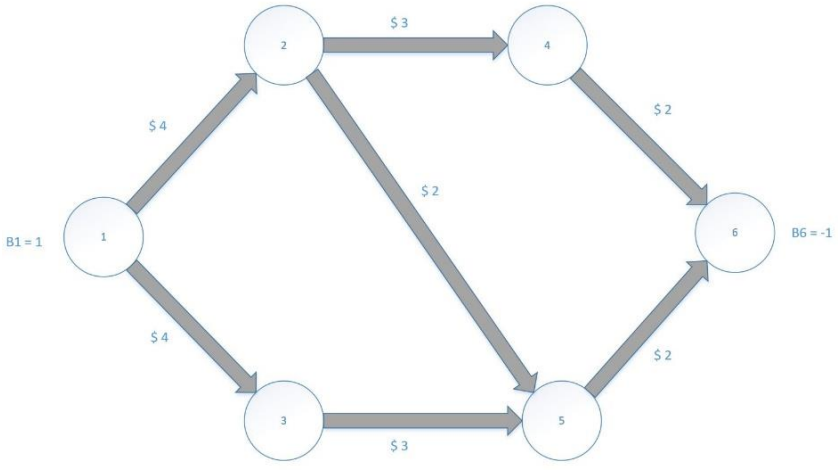


Figure 6. Initial network example 1

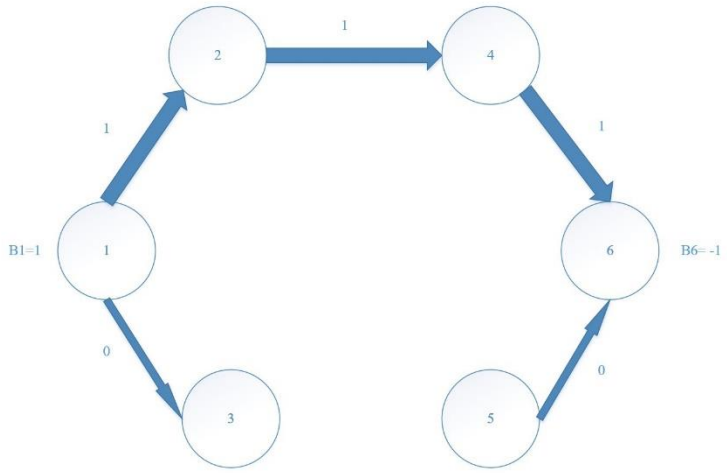


Figure 7. First step to solve the shortest path problem in example 1

In Step 2, the complementary slackness theorem on the dual constraints is applied. It means to solve the following system of equations for the primal basic variables. The system has one redundant W variable that can be set equal to zero to solve for the other variables. If W_1=0 the values of the W's are computed below and depicted in Figure 8.

$$W_1 - W_2 = C_{12} = 4$$

$$W_2 - W_4 = C_{24} = 3$$

$$W_4 - W_6 = C_{46} = 2$$

$$W_1 - W_3 = C_{13} = 3$$

$$W_5 - W_6 = C_{56} = 2$$

Now, the optimality of the non-basic variables is checked by using the dual constraints and the complementary slackness theorem.

$\bar{C}_{25}$ = [W_2 - W_5 - C_{25}] → [-4 - (-7) - 2] → 1 (Violates optimality condition)

$\bar{C}_{35}$ = [W_3 - W_5 - C_{35}] → [-3 - (-7) -3] → 1 (Violates optimality condition)

Since $\bar{C}_{25}$ and $\bar{C}_{35}$ violate the optimality conditions with a value 1, arbitrarily $\bar{C}_{25}$ is taken. Step 3 of the NSM is also shown in Figure 8. X_{25} will enter the basis with an

allocation of Θ equal to 1 because beyond this value, arcs (2,4) and (4,6) will become negative. X_{46} will leave the basis as shown by the cycle formed by arcs X_{24}, X_{46}, X_{25} and X_{56}.

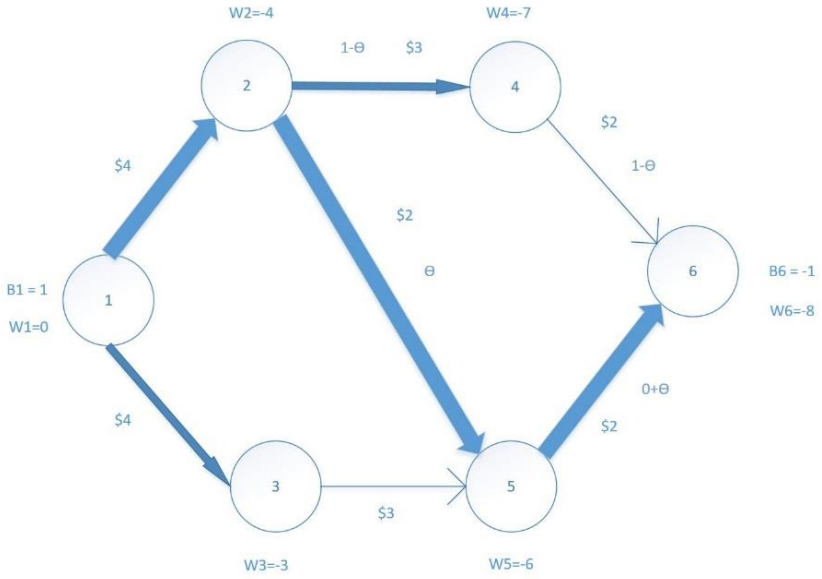


Figure 8. Steps 2 and 3 NSM to solve the shortest path problem in example 1

The lighter arcs in Figure 8 indicate basic variables at the lower level of 0 and the thicker arcs represent the remaining basic variables. The new basic arcs are (1,2) (2,4) (2,5) (1,3) and (5,6). A new iteration of the NSM starts by repeating step 2. The new W values are:

$$W_2 - W_5 = C_{25} = 2$$

$$W_2 - W_4 = C_{24} = 3$$

$$W_5 - W_6 = C_{56} = 2$$

$$W_1 - W_3 = C_{13} = 3$$

$$W_1 - W_2 = C_{12} = 4$$

Now, the optimality of the non-basic variables is checked by using the dual constraints and the complementary slackness theorem.

$\bar{C}_{35}$ = [W_3 - W_5 – C_{35}] → [-3 - (-6) -3] → 0 (Satisfies optimality condition)

$\bar{C}_{46}$ = [W_4 - W_6 – C_{46}] → [-7 - (-8) -2] → -1 (Satisfies optimality condition)

Since both non-basic variables satisfy the optimality condition, the algorithm stops. Figure 9 shows the optimal shortest path solution.

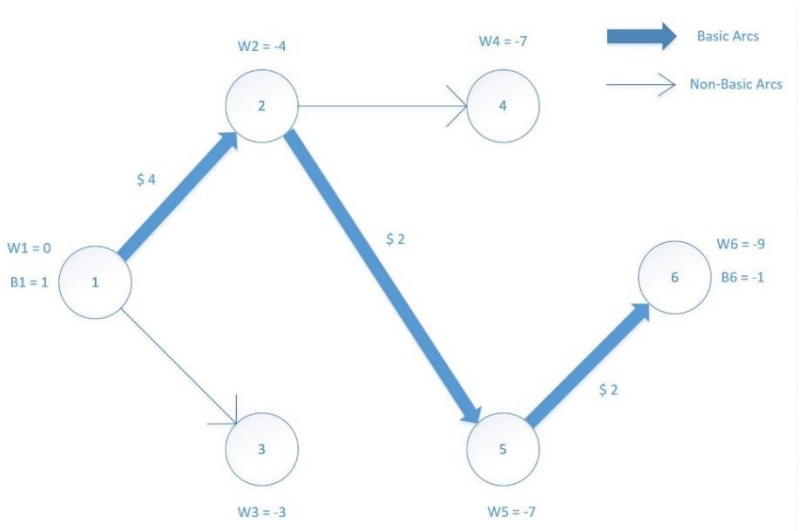


Figure 9. Optimal solution for the shortest path problem in example 1

4.3.3 Example 2: solving a Transshipment Problem using the NSM

Figure 10 presents the problem. Node 1 and 3 are supply nodes, while node 2 is a transshipment node and node 4 and 5 are demand nodes. The B values indicate the amount of demand (-) and/or supply (+) on each node. The cost associated with each arc are along the arrows. Figure 11 shows the status of the algorithm after the first step is completed.

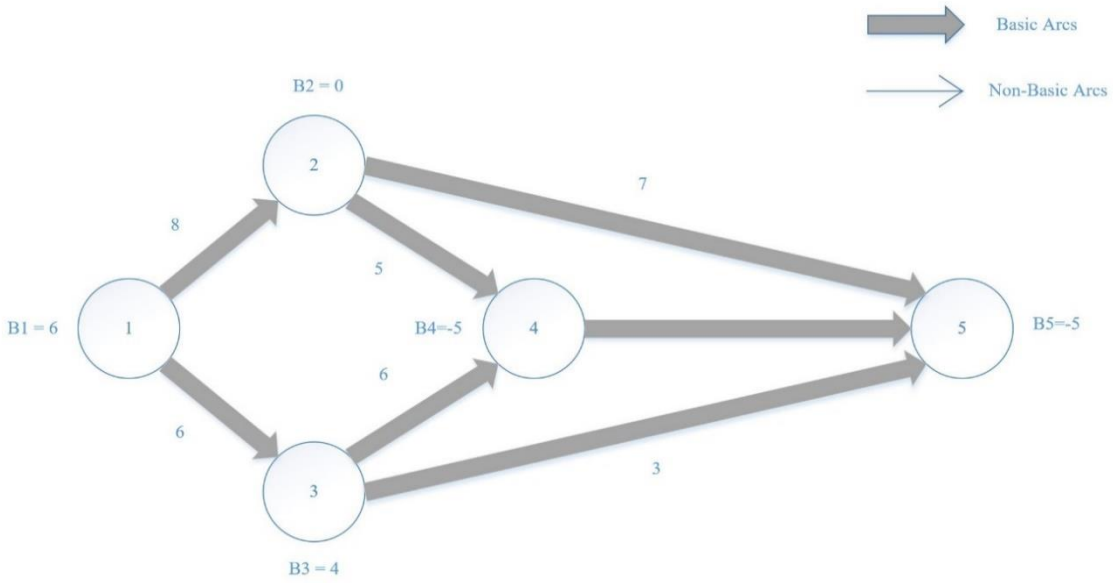


Figure 10. Initial network example 2

In the first step, the arcs (1,2) (2,4) (3,4) and (3,5) are arbitrarily selected as the bfs. Figure 11 has the allocated values for these basic variables (i.e. arcs). See values after the x's. Figure 11 also shows the W's resulting after solving the system of equations in step 2 of the NSM. The system results from applying the complementary slackness theorem on the dual constraints. The system has one redundant W variable that can be set equal to zero to solve for the other variables. If W_5=0 the values of the W's are the ones listed below and depicted in Figure 11.

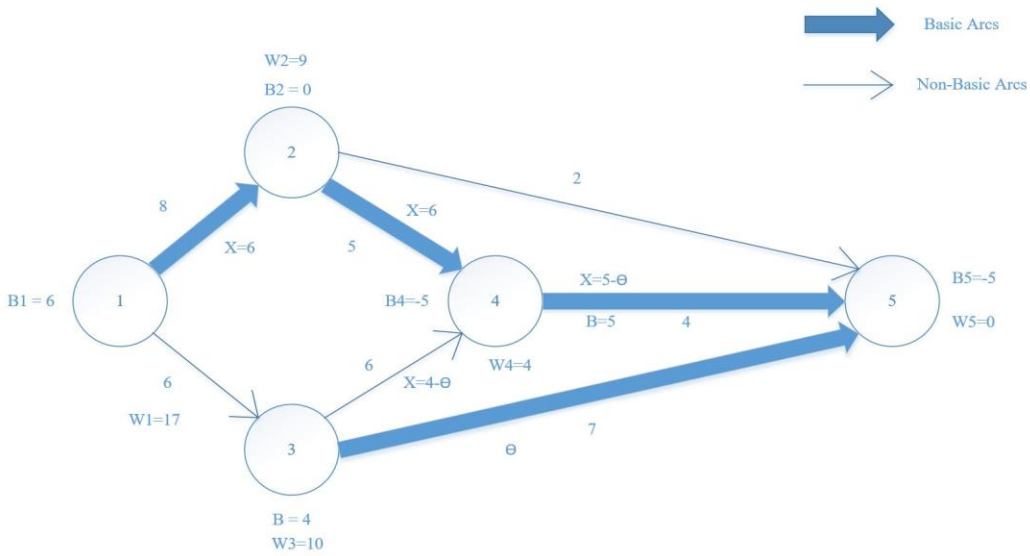


Figure 11. Network simplex method steps for the first iteration in example 2

$$W_1 - W_2 = C_{12} = 8$$

$$W_2 - W_4 = C_{24} = 5$$

$$W_3 - W_4 = C_{34} = 6$$

$$W_4 - W_5 = C_{45} = 4$$

Now, the optimality of the non-basic variables is checked by using the values of the dual constraints and the complementary slackness theorem.

$\bar{C}_{25} = [W_2 - W_5 - C_{25}] \rightarrow [9 - 0 - 7] \rightarrow 2$ (Violates optimality condition)

$\bar{C}_{13} = [W_1 - W_3 - C_{13}] \rightarrow [17 - 10 - 1] \rightarrow 6$ (Violates optimality condition)

$\bar{C}_{35} = [W_3 - W_5 - C_{35}] \rightarrow [10 - 0 - 3] \rightarrow 7$ (Violates optimality condition)

$\bar{C}_{35}$ is the maximum violating arc and it enters to the bfs. In Step 3 of the NSM, an allocation of $\Theta = 4$ is given to the new entering arc 3-5 because in the cycle formed by arcs X_{34}, X_{45}, and X_{35}, a value larger than 4 will give a negative value to the basic variable X_{34}. The variable X_{34} then leaves the basis. Figure 11 shows with bolded color the current bfs after completing one iteration of the steps in the NSM. The second iteration starts with

arcs (1,2), (2,4), (4,5) and (3,5) as basic variables. Figure 12 shows the values for the W's computed in the second step at the second iteration. They result from solving the equations below Figure 12, setting W_5=0 and using the complementary slackness theorem again.

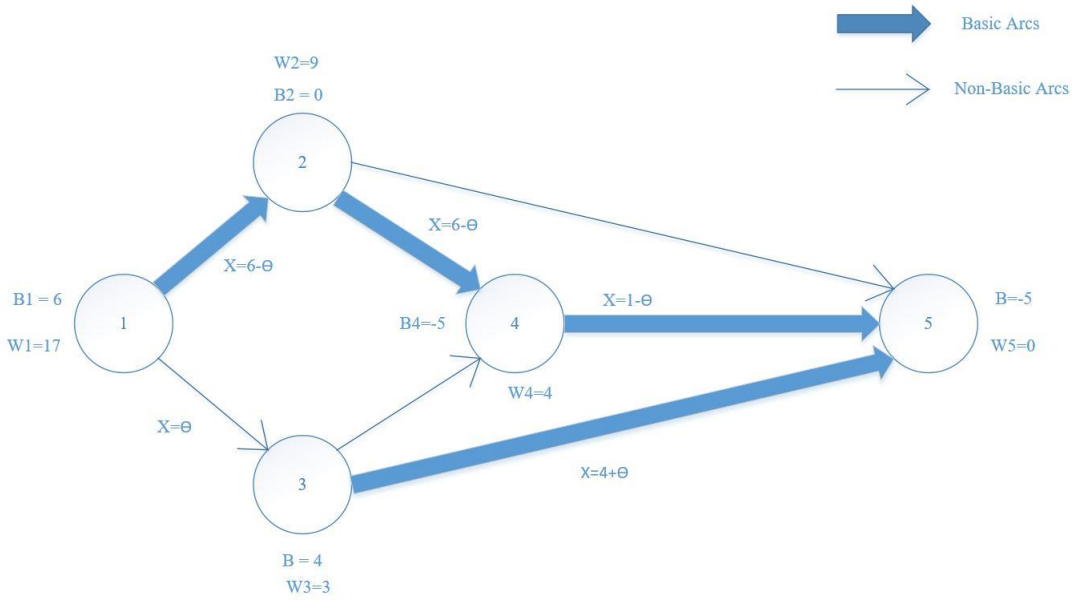


Figure 12. Second iteration of the network simplex method for example 2

$$W_1 - W_2 = C_{12} = 8$$

$$W_2 - W_4 = C_{24} = 5$$

$$W_4 - W_5 = C_{45} = 4$$

$$W_3 - W_5 = C_{35} = 3$$

Now, the optimality of the non-basic variables is checked.

$\bar{C}_{34} = [\, W_3 - W_4 - C_{34}\,] \rightarrow [\, 3 - 4 - 3] = -4$ (Satisfies optimality condition)

$\bar{C}_{13} = [W_1 - W_3 - C_{ij}] \rightarrow [17 - 3 - 6] = 8$ (Violates optimality condition)

$\bar{C}_{25} = [W_2 - W_5 - C_{ij}\,] \rightarrow [9 - 0 - 7] = 2$ (Violates optimality condition)

$\bar{C}_{13}$ is the maximum violating arc and it enters into the bfs. In step 3, an allocation of Θ=1 is given for the new entering arc 1-3 because in the cycle formed by arcs X_{12}, X_{24},

X_{45}, X_{13} and X_{35} it is the maximum value that permits the arc 4-5 to stay above its lower value of zero. Then X_{45} leaves the basis. The iterations of the NSM continue until the optimum is achieved. In example 2, the optimum is achieved in 4 iterations. Figure 9 shows the optimal network.

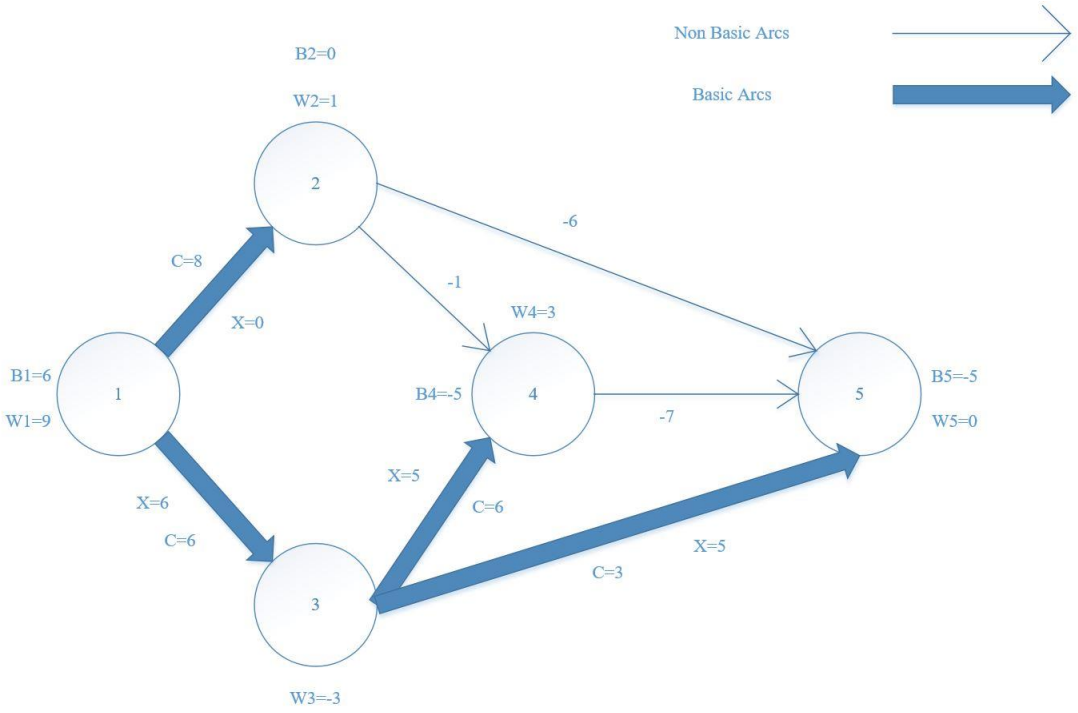


Figure 13. Optimal network for example 2 after four iterations

4.4 Modeling the DFLP and the SDFLP as a linear network problem with AMPL

AMPL stands for Advanced Mathematical Programming Language. It is an algebraic modelling language to describe and solve highly complex problems occurring in large scale optimization and scheduling-type problems. Network linear programs can be modeled in AMPL using standard AMPL formulations that include definition of sets, parameters, variables, objective function and constraints. AMPL also permits network models to be described more directly in terms of their network structure using what is known as *node and arc declarations.* The advantage of modeling the problem using *node*

and arc declarations is that AMPL automatically communicates the network structure to the solver and any special network algorithms in the solver are applied automatically. (Fourer, 2003).

In this thesis, the solver selected to work with AMPL is CPLEX. CPLEX incorporates an optional heuristic procedure that identifies "pure network" constraints in a linear program. The CPLEX procedure looks for these constraints and if the model has many of them CPLEX applies a fast network simplex algorithm. If the CPLEX solver finds non-network constraints, CPLEX uses the network solution as a start for solving the problem by the general primal or dual simplex algorithm. The optional heuristic is active by default but the user can suppress it or force its use in all cases. CPLEX's network simplex algorithm can achieve dramatic reductions in optimization time for "pure" network linear programs defined entirely in terms of *node and arc declarations* (Fourer,2003).

The *node-and-arc declaration* makes it easy to define a linear program for a network that has several different kinds of nodes and arcs. The DFLP problem in this thesis, modeled as presented in Section 2.3, is solved in AMPL using *node and arc declarations*. Consequently, the various steps and iterations that such formulation undergoes to find a solution are the ones described and depicted in the Section 4.3 Network Simplex Algorithm for solving DFLP modeled as a network problem. Examples 4.3.2 & 4.3.3 illustrated the procedure AMPL/CPLEX follows by using a small-sized problem. Because the models for solving the DFLP with budget constraint and the SDFLP model presented in sections 2.4 and 2.5 have additional constraints they are considered as variants to the regular shortest path problem. Thus, they are not solved with the simplex network algorithm.

The AMPL model and a skeleton sample data file used for solving the DFLP proposed in Section 2.3 using *node and arc declarations* are presented in Appendix C and D, respectively. The AMPL model and a skeleton sample data file for solving the DFLP with budget proposed in Section 2.4 are in Appendix E and F. The AMPL model and a skeleton sample data file for solving the SDFLP proposed in Section 2.5 are presented in Appendix G and H.

5. NUMERICAL RESULTS

In this chapter, the DFLP *Parallel Shortest Path (PSP)* variation of the Dijkstra algorithm and the *Linear Network Model (LNM)* methodology are compared and contrasted to several heuristic approaches proposed by previous authors. The comparison is done in terms of computational time and overall cost of the solution.

5.1 DFLP Datasets

Balakrishnan & Cheng (2000) generated DFLP instances that the research community have been using to test their proposed methodologies. The instances have 6, 15 and 30 departments and 5 and 10-time periods. Under each combination of department and period, there are eight different problems that differ by the matrixes of flow between departments and distance between locations. In this thesis, all problems with 6, 15 and 30 departments and 5-time periods were selected and tested. The fixed time horizons of 5 periods seemed to make more practical sense. Due to volatility in the economy, fixed periods of 10-years length seemed less practically implementable. In addition, Moslemipour & Lee (2012) present a randomly generated problem with 12 departments and 5-time periods. In this thesis, this 12 departments instance is also used for experimentation.

5.2 SDFLP Datasets

Since there is no library of instances for the SDFLP, 4 problems were generated with Excel inspired by the methodology Balakrishnan et al (1992) followed to generate instances for the DFLP. The instances considered five-time periods, six departments and locations and three scenarios for the market that will affect the matrices of flows between departments and the vectors of costs of relocating the departments. The flows between

departments necessary to compute the material handling cost (MHC) and to obtain the values for the $C_{Sit,k(t+1)}$ parameters were randomly generated from a uniform distribution (U[a,b]) with parameters a=100 and b=200. For each problem, the total material flow between departments was kept constant over the time periods. Then, the flows for three departments were increased by a factor of 5 in three departments (departments 1,4, and 6). In this way, flow dominance from some departments would motivate relocations and the flow matrixes end asymmetric. The distance (feet) matrix between locations was about the same order of magnitude than the ones provided from the library (*d6l5*) in Balakrishnan & Cheng (2000).

The vectors of costs for relocating each department, named uRC in the codes, were generated randomly from a uniform distribution (U[a,b]) with a=100 and b=500 in each of the three scenarios considered. The procedure to generate flow matrices and vectors of cost of relocating each department was repeated for the 4 problems, and for the 3 scenarios and 5-time periods in each.

5.3 DFLP experimental Setting

The number of different DFLP problems studied is 25 (8 problems with 6, 15 and 30 departments and 1 problem with 12 departments). Each DFLP problem was also run 5 times. The multiple runs are necessary to see the variability of the results to selecting different sets of layouts or permutations to include in the DFLP network.

The number of runs equals to 125 (25*5) for the variant of no-sorting the layouts randomly generated to input to the network. This variant is labeled as *no-sorting* through this document. A total of 120 (24*5) runs were done for the sorting layouts variant (since the 12 instances was not studied). This variant is labeled as *sorting* through this document.

Since this thesis compares two methodologies, *PSP* and *LNM*, 250 runs were executed for the *non-sorting* variant and 240 for the *sorting* variant.

The first *PSP* algorithm implemented was the one that does not sort the permutations or layouts to enter in the network nodes based on its MHC. This *PSP* was implemented in C language by Kolla (2015) under the supervision of Dr. Apan Qasem and Dr. Clara Novoa. This implementation had some OpenMP calls. Later, Dr. Apan Qasem developed an implementation of the *PSP* that sorts the permutations to enter to the network nodes in each year based on the annual MHC incurred.

The first C implementation (i.e. the *no-sorting* variant) was run using the *Stampede* and *Maverick* clusters from the Texas Advanced Computing Center (TACC), https://portal.tacc.utexas.edu/. The cluster that was used the most was *Stampede* (100 runs). Each one of the 6400 *Stampede* nodes runs under the CentOS 6.4 Operating system and has the following characteristics: CPU Intel Xeon E5-2680 v2 Ivy Bridge, 2.80 GHz, 20 CPU's/node, 12.8 GB memory/core. Also, 25 runs were done in the *Maverick* cluster equipped with 132 nodes each one with the following characteristics: CentOS, Dell C8220, Intel PQI, C610 Chipset, 2/8 Xeon E5-2680 2.7GHz (turbo, 3.5) 1/61 Xeon Phi SE10P 1.1GHz, 10 CPU cores with 32 GB 8x4G DDR3-1600 MHZ 8GB GDDR5. One input to the *PSP* implementation was the number of permutations to randomly generate for each period. This number was set to 85,000. As shown in Figure 14, the C program uses flow and distances data from the instances, generates a number of permutations per period equal to min (n!, 85,000), assign them to the nodes in the *PSP* network, computes MHC at the nodes or vertices and RC at the edges, and performs the *PSP*, a variant of the Dijkstra algorithm described in detail in Chapter 4.

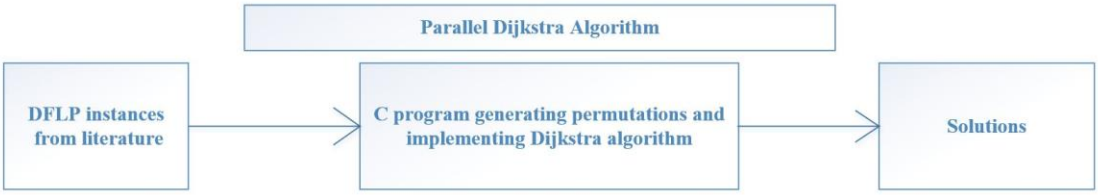


Figure 14. Flow chart to solve the DFLP using the *PSP* implementation

For the experimentation under *sorting* (i.e. the *sorting* variant), a larger memory machine named CAPI was used. CAPI (Coherent Accelerator Processor Interface) is available at the Computer Science department at Texas State. The machine was donated by IBM. CAPI runs under CentOS. It has 160 CPU's, min speed 2.061 Ghz and max speed 3.690 Ghz. CAPI has an additional functionality for PCIe slots on CAPI enabled systems. It uses 16 PCIe slots and is layered on top of PCIe Gen 3. CAPI port is determined by the underlying PCIe 3.0 x16 technology, peaking at ca 16 GB/s, bidirectional. PCI Express 3.0's 8 GT/s bit rate effectively delivers 985 MB/s per lane, nearly doubling the lane bandwidth relative to PCI Express 2.0 Latest versions of CAPI have been consistently updated. They have more capabilities and can be used in the future research. A comparison of the computational environments used in this thesis is in Appendix M.

To solve the problems under the *LNM*, the *PSP* implementation in C was modified by deleting the variant of the Dijkstra procedure and introducing additional lines in C code to print the input data files for AMPL following specific syntax required by AMPL under the *node and arc declaration* style. The C modified implementation was asked to randomly generate only 2,000 permutations to feed the network nodes in each year. This relatively small number of permutations was used, if compared to 85,000 used in the *PSP* implementation, because when solving larger input data problems there were computer

memory issues and AMPL was unable to provide a solution. The steps to solve under the *LNM* are presented in Figure 15.

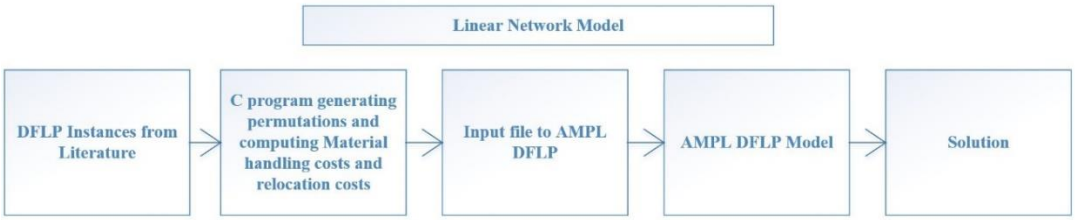


Figure 15. Flow chart to solve the DFLP under the *Linear Network Model*

To generate the *LNM* solutions, the 8 problems with 6 departments were run in a personal laptop, Dell studio XPS 1645, Core i7-720QM 1.6 GHz Processor, windows 7 professional, 64-bit OS, 4GB Ram. The rest of the problems were run in the *SOLAR lab* at Texas State University, http://www.engineering.txstate.edu/Facilities/IE-labs/RFM-4244.html, using a Dell Optiplex 5040, Intel Core i5-6500 CPU @3.20 GHZ, 16GB Ram, Quad-Core (i.e. 4 cores), 64 Bit OS, Windows 10 Enterprise.

5.4 SDFLP experimental setting

The data generated in Excel for each of the 4 SDFLP problems studied was saved in text files that were fed to a C program which is another variation of the one described in the previous section. It generates 6! random layouts per period and computes MHC and RC for each scenario and period. The C program was run 20 times (4 problems, 5 runs in each) in *Leap*, a high-performance computing cluster at Texas State University managed by the Division of Information Technology. *Leap*, http://www.vpit.txstate.edu/rc/leap.html has 120, Intel Xeon E5-2680v4 nodes each one with 28 CPU cores running at 2.4 GHz speed, 128 GB memory running under CentOS. The output of the C program is the input data file to the AMPL model created to solve the SDFLP problem. Figure 16 presents the steps described above.

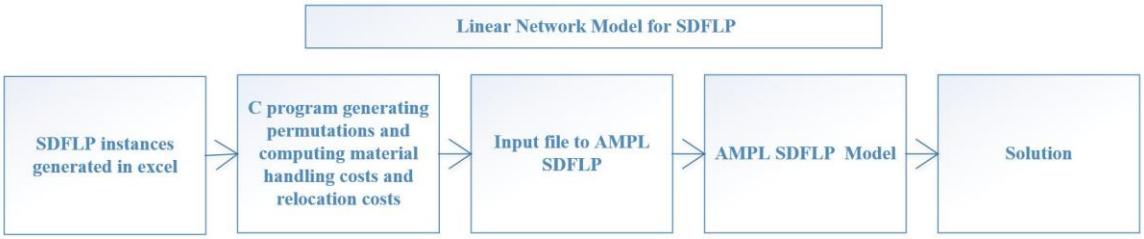


Figure 16. Flow chart to solve the SDFLP under the *Linear Network Model*

5.5 DFLP cost results *PSP* vs. *LNM*

5.5.1 *No-sorting*

Final costs for both *PSP* and *LNM* were collected and averaged over the five runs done on each problem. The standard deviation was also computed. Tables 7-13 show the resulting costs for the variant in which the permutations were randomly generated and not sorted. In the tables, the abbreviation "PN" stands for problem number, the abbreviation "Std Dev" stands for standard deviation and the abbreviation "R" stands for run. The costs for the 6-departments problems are the same for *PSP* and *LNM* while for the 12, 15 and 30 departments problems *PSP* gave slightly better results. However, the *LNM* gave very competitive results considering that this method has considerably less permutations in the network. *LNM* costs are no more that 2.17% above those from *PSP*.

Table 7. Cost Results for DFLP *PSP* 6 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	106419	106419	106419	106419	106419	106419	0
2	105341	105341	105341	105341	105341	105341	0
3	102989	102989	102989	102989	102989	102989	0
4	106399	106399	106399	106399	106399	106399	0
5	105628	105628	105628	105628	105628	105628	0
6	103985	103985	103985	103985	103985	103985	0
7	106439	106439	106439	106439	106439	106439	0
8	103771	103771	103771	103771	103771	103771	0

Table 8. Cost Results for DFLP *LNM* 6 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	106419	106419	106419	106419	106419	106419	0
2	105341	105341	105341	105341	105341	105341	0
3	102989	102989	102989	102989	102989	102989	0
4	106399	106399	106399	106399	106399	106399	0
5	105628	105628	105628	105628	105628	105628	0
6	103985	103985	103985	103985	103985	103985	0
7	106439	106439	106439	106439	106439	106439	0
8	103771	103771	103771	103771	103771	103771	0

Table 9. Cost Results for DFLP *PSP* and *LNM 12* Departments - *No-Sorting*

PN		R1	R2	R3	R4	R5
1	*PSP*	1273487	1252704	1278564	1278994	1259654
1	*LNM*	1501492	1501176	1418920	1455610	1350250
			Average	Std Dev		
			1268681	11873		
			1431489	63759.6		

Table 10. Cost Results for DFLP *PSP* for 15 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	502383	503061	501022	501945	499324	501,547.0	1446.3
2	502968	501203	502653	503398	504767	502,997.8	1287.6
3	507007	505240	506566	507488	508116	506,883.4	1084.2
4	503146	501079	500181	502232	498535	501,034.6	1793.8
5	500253	499790	501644	500077	499796	500,312.0	770.0
6	501869	503221	501073	502880	503385	502,485.6	985.0
7	504497	506293	502415	502871	504423	504,099.8	1534.7
8	505986	505682	508692	506986	509664	507,402.0	1726.1

Table 11. Cost Results for DFLP *LNM* for 15 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	509933	511825	512736	511302	509640	511,087.2	1,297.7
2	515116	513235	505768	516033	513011	512,632.6	4,042.2
3	514849	519094	516696	511302	514118	515,211.8	2,910.7
4	509335	508651	507627	506729	508525	508,173.4	1,010.7
5	508626	507496	512751	512867	513982	511,144.4	2,883.3

Table 11. (Continued)							
6	509046	513107	511075	513996	510268	511,498.4	2,033.0
7	513470	514088	514020	513943	514864	514,077.0	502.3
8	519291	517236	517310	517884	517169	517,778.0	892.2

Table 12. Cost Results for DFLP *PSP* for 30 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	601677	602742	603472	603722	604214	603,165.4	987.2
2	597363	600518	600037	601620	602501	600,407.8	1953.9
3	609946	607188	609449	604782	607237	607,720.4	2067.2
4	600429	602334	600736	603536	599272	601,261.4	1677.1
5	587114	589020	594161	591567	590409	590,454.2	2654.9
6	599630	596776	597768	597645	597601	597,884.0	1052.0
7	597888	589807	597501	598014	596290	595,900.0	3473.5
8	603205	606773	604466	605378	600036	603,971.6	2556.7

Table 13. Cost Results for DFLP *LNM* for 30 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	610863	609194	608418	609015	609364	609,370.8	907.2
2	608969	602084	609999	608834	603844	606,746.0	3,536.9
3	611281	609360	609949	615224	613051	611,773.0	2,394.5
4	612074	602438	609192	604111	608623	607,287.6	3,934.8
5	597523	601774	600448	595799	602353	599,579.4	2,818.9
6	602212	606536	601016	603545	603283	603,318.4	2,057.3
7	600228	602175	600160	603011	603538	601,822.4	1,564.1
8	612718	617314	614259	618330	611595	614,843.2	2,901.3

5.5.2 Sorting

To determine if generating a larger number *X* of permutations per year and filtering the best first *Y* in terms of MHC improve the solutions from the two solution approaches (*PSP* and *LNM*), a total of *X* = 400,000 permutations was generated for *PSP* and X= 85,000 for *LNM*. The material handling costs for each year and permutation was computed. The *Y*= 50,000 permutations having the lowest costs in each year were selected as input to the *PSP*. The Y=2,000 permutations with the lowest cost in each year were selected as input to the *LNM* approach. This selection of layouts is done using the "Quick sort algorithm".

Quick Sort Algorithm sorts data by dividing large arrays into 2 smaller arrays by utilizing a divide and conquer strategy. Quick Sort Algorithm Sorts the layouts in a particular year based on the following steps:

- Picking a "Pivot" Element
- "Partitioning" the array into 3 parts. In the first part, all elements are less than the pivot. The second part is the pivot itself (only one element). In the third part, all the elements are greater than or equal to the pivot.
- Applying recursively the Quick Sort Algorithm to the first and the third parts of the array.

Final costs for *PSP* and *LNM* were collected and averaged over the five runs done to each problem. The standard deviation was also computed. Tables 14-19 shows the resulting costs; the abbreviations used in these tables are for problem number (PN), run (R) and standard deviation (Std). The costs for the 6-departments problems for *PSP* and *LNM* are the same while for the 15 and 30 department problem *LNM* gave results not more than 3.40% above those in *PSP*.

Table 14. Cost Results for DFLP *PSP* for 6 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	106419	106419	106419	106419	106419	106419	0
2	105341	105341	105341	105341	105341	105341	0
3	102989	102989	102989	102989	102989	102989	0
4	106399	106399	106399	106399	106399	106399	0
5	105628	105628	105628	105628	105628	105628	0
6	103985	103985	103985	103985	103985	103985	0
7	106439	106439	106439	106439	106439	106439	0
8	103771	103771	103771	103771	103771	103771	0

Table 15. Cost Results for DFLP *LNM* for 6 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	106419	106419	106419	106419	106419	106,419	0

Table 15. (Continued)							
2	105341	105341	105341	105341	105341	105,341	0
3	102989	102989	102989	102989	102989	102,989	0
4	106399	106399	106399	106399	106399	106,399	0
5	105628	105628	105628	105628	105628	105,628	0
6	103985	103985	103985	103985	103985	103,985	0
7	106439	106439	106439	106439	106439	106,439	0
8	103771	103771	103771	103771	103771	103,771	0

Table 16. Cost Results for DFLP *PSP* 15 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	500507	495473	499358	498621	497406	498,273.0	1,929
2	497275	500387	501017	498662	499142	499,296.6	1,472
3	499744	499873	499660	499660	501395	500,066.4	748
4	498665	498251	498707	500457	497507	498,717.4	1,085
5	498733	499223	499512	499806	499796	499,414.0	450
6	499405	499556	499694	497515	500596	499,353.2	1,127
7	502451	500889	500052	500109	502345	501,169.2	1,170
8	505986	505409	505855	506986	504570	505,761.2	881

Table 17. Cost Results for DFLP *LNM* for 15 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	502383	503061	501022	501945	499324	501,547.0	1,446
2	502968	501203	502653	503398	504767	502,997.8	1,288
3	507007	505240	506566	504767	509312	506,578.4	1,784
4	503146	501079	500181	502232	498535	501,034.6	1,794
5	500253	499790	501644	500077	499796	500,312.0	770
6	501869	503221	501073	502880	503385	502,485.6	985
7	504497	506293	502415	502871	504423	504,099.8	1,535
8	505986	505682	508692	506986	509864	507,442.0	1,793

Table 18. Cost Results for DFLP *PSP* for 30 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	601577	605578	604070	600346	604596	603233.4	2187
2	603036	603455	603250	604517	602403	603332.2	771
3	605506	606976	606469	603952	606011	605782.8	1159
4	598641	598470	597683	595931	599272	597999.4	1288
5	585817	589020	585213	588691	586792	587106.6	1697
6	595499	595757	594274	595855	593583	594993.6	1011
7	597802	594831	596663	597906	598693	597179.0	1499
8	601448	602638	601616	602577	600036	601663.0	1059

Table 19. Cost Results for DFLP *LNM* for 30 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	623712	624958	625140	624347	624161	624463.6	586
2	606868	607110	606678	608297	605593	606909.2	969
3	605593	611584	611584	614039	617973	612154.6	4503
4	600429	602438	602079	604389	599272	601721.4	1964
5	587114	589489	594161	591567	590409	590548.0	2599
6	599819	598573	598124	597645	598853	598602.8	820
7	600898	595211	599727	603858	599984	599935.6	3110
8	603205	606773	598897	605378	600036	602857.8	3371

5.6 DFLP computational time results *PSP* vs. *LNM*

5.6.1 *No-sorting*

Data collected for the computational time for the two solution methodologies (*PSP* and *LNM*) are in Tables 20-27. For *PSP*, the time measured is the one for generating the layouts, computing MHC and RC and solving the variation of the Dijkstra algorithm (i.e. all steps in Figure 14). For the *LNM*, the time measured is only the one for solving the *LNM* in AMPL. The time for generating the layouts and computing the MHC and RC is very negligible since the C code has several of these functions parallelized. In the 6-department problem, the computational time for the *PSP* is very small. For 12, 15, and 30 departments, the *LNM* gave faster computational times in several cases. However, in the 15 department the first and second problem were solved faster with the *PSP*. *LNM* ran faster than *PSP* in 75 runs. The *PSP* performed better in the remaining 50 runs. Appendix I has samples of outputs generated by the *PSP* implementation in C and the *LNM* implementation in AMPL. The outputs show best solutions found and computational times for some of the problems run under the *no-sorting* version.

Table 20. Computational Time (Seconds) for *PSP* for 6 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	0.87	1.19	0.99	0.98	1.11	1.03	0.13
2	0.13	0.13	0.13	0.13	0.13	0.13	0.00

Table 20. (Continued)							
3	0.21	0.13	0.14	0.13	0.12	0.15	0.04
4	0.98	0.94	0.94	0.89	0.87	0.92	0.04
5	4.87	0.40	1.61	0.14	5.19	2.44	2.43
6	0.14	0.30	0.24	0.13	0.13	0.19	0.08
7	0.31	0.31	0.13	0.13	0.13	0.20	0.10
8	0.21	0.13	0.13	0.13	0.13	0.15	0.04

Table 21. Computational Time (Seconds) for *LNM* for 6 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	5.23	4.98	4.95	5.03	5.06	5.05	0.11
2	10.89	11.20	11.20	11.04	11.39	11.14	0.19
3	10.78	11.17	11.26	10.92	11.59	11.14	0.32
4	10.78	11.23	11.47	11.48	10.98	11.19	0.31
5	11.25	10.76	10.72	11.61	11.20	11.11	0.37
6	10.99	10.89	11.50	11.15	11.29	11.17	0.24
7	11.43	11.34	11.17	10.95	11.33	11.24	0.19
8	11.36	11.34	11.67	11.00	11.98	11.47	0.37

Table 22. Computational Time (Seconds) Results for *PSP* for 12 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	94.86	97.21	95.91	99.07	95.48	96.50	1.67

Table 23. Computational Time (Seconds) for *LNM* for 12 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	65.66	68.27	65.81	59.84	63.27	64.57	3.18

Table 24. Computational Time (Seconds) for *PSP* for 15 Departments - *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	35.70	35.65	37.25	38.99	34.73	36.46	1.68
2	96.56	100.06	92.44	98.32	99.69	97.41	3.10
3	400.94	396.36	396.17	395.33	395.05	396.77	2.40
4	394.68	394.61	398.26	397.36	395.39	396.06	1.66
5	394.30	394.79	398.10	399.61	395.36	396.43	2.31
6	395.56	395.41	394.58	399.57	395.19	396.06	1.99
7	396.99	394.97	394.51	394.82	397.53	395.76	1.39
8	399.96	334.96	395.02	396.05	394.25	384.05	27.53

Table 25. Computational Time (Seconds) for *LNM* for 15 Departments – *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	61.23	64.16	75.58	61.38	60.92	64.65	6.25
2	62.00	59.95	62.20	60.61	60.95	61.14	0.95
3	64.59	65.42	60.69	59.91	59.80	62.08	2.71
4	59.92	60.33	60.34	59.95	61.02	60.31	0.44
5	60.39	59.52	60.38	60.95	60.94	60.43	0.59
6	59.83	60.42	61.23	61.28	61.67	60.89	0.75
7	60.03	59.63	59.80	60.82	61.17	60.29	0.67
8	59.86	59.61	60.70	60.56	58.86	59.92	0.75

Table 26. Computational Time (Seconds) for *PSP* for 30 Departments – *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	455.41	457.85	456.14	456.88	458.75	457.01	1.33
2	457.11	453.77	456.55	453.99	465.44	457.37	4.75
3	460.53	456.75	455.43	455.13	454.18	456.40	2.49
4	105.30	110.02	101.16	110.31	104.02	106.16	3.95
5	104.05	107.96	105.31	113.57	107.87	107.75	3.66
6	104.09	111.72	102.79	112.61	108.12	107.86	4.40
7	109.71	103.15	113.62	116.06	113.46	111.20	5.04
8	112.06	106.05	101.63	108.68	107.17	107.12	3.81

Table 27. Computational Time (Seconds) for *LNM* for 30 Departments – *No-Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	62.17	60.27	60.48	59.53	79.26	64.34	8.40
2	75.18	76.16	75.01	55.06	78.09	71.90	9.49
3	60.11	60.83	59.48	62.56	60.83	60.76	1.15
4	60.31	59.09	60.45	59.67	60.59	60.03	0.63
5	59.11	60.25	60.45	60.30	60.19	60.06	0.54
6	76.25	77.98	79.09	78.28	76.47	77.62	1.22
7	60.58	60.83	60.36	59.50	59.34	60.12	0.66
8	77.75	75.55	76.52	77.52	76.94	76.86	0.87

5.6.2 *Sorting*

For the *sorting* variant, the parallelization in the *Shortest Path* algorithm was removed and consequently the algorithm is notated as *SP* in the remainder of this

document. Computational times for the *SP* method, including the times for *sorting* the 400,000 generated permutations and filtering the best 50,000 in each year, are in Tables 28-33. These tables also show the time for solving the *LNM* in AMPL. For the *LNM*, the times reported do not include the time for *sorting* the 85,000 generated permutations and filtering the best 2,000 in each year.

It is important to mention that for the *SP* method, two times were collected. The first one is the entire time the algorithm takes to generate the permutations, sort them, and execute the variation of the Dijkstra algorithm (Tables 28-33). The second one is the time to execute just the variation of the Dijkstra algorithm. Out of the entire time collected, the variation of the Dijkstra algorithm accounts for a maximum time of 9.939%. Tables with the time to execute just the variant of the Dijkstra's algorithm are in Appendixes N, O, and P. Appendix N presents the differences in times for doing the variation of the Dijkstra's algorithm for SP *sorting* vs. *PSP no sorting*, Appendix O presents the times *SP sorting* took to perform the variation of the Dijkstra in each of the 5 runs done in each problem, the average and standard deviation. Appendix P presents the differences in times for doing the variation of the Dijkstra's algorithm for *SP sorting* vs. *LNM sorting*. Appendix Q presents a graph that shows Problem 1 average computational times over the 5 runs done in the 6, 15, and 30 departments cases. The graph lets to appreciate the trends on the computational times for all the methods studied. *SP* and *LNM* showed similar execution times if looking just at the portion of time spent on doing the variation of the Dijkstra algorithm for *SP-sorting* 50,000 layouts and the network simplex method for *LNM-sorting* 2,000 layouts.

If analyzing results in Tables 28-33, in the 6-department problems, the *SP* computational time is very small and smaller than the one for *LNM*. For 15, and 30 departments the *LNM* gave faster computational times. The *LNM* ran faster than *SP* in 80 runs while the *SP* performed better in 40 runs. The number of runs for the sorting case was 120 since the Moslemipour instance of size 12 was not run for the *sorting* case.

Table 28. Computational Time (Seconds) for *SP* 6 Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	0.13	0.12	0.12	0.12	0.12	0.122	0.004
2	0.13	0.12	0.12	0.12	0.12	0.122	0.004
3	0.13	0.12	0.12	0.12	0.12	0.122	0.004
4	0.13	0.12	0.12	0.12	0.12	0.122	0.004
5	0.13	0.12	0.12	0.12	0.12	0.122	0.004
6	0.13	0.12	0.12	0.12	0.12	0.122	0.004
7	0.13	0.12	0.12	0.12	0.12	0.122	0.004
8	0.13	0.12	0.12	0.12	0.12	0.122	0.004

Table 29. Computational Time (Seconds) for *LNM* 6 Departments – *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	5.67	5.30	5.17	5.10	5.63	5.374	0.262
2	5.55	5.58	5.24	5.28	5.19	5.368	0.183
3	5.22	5.60	5.50	5.66	5.86	5.568	0.235
4	5.31	5.21	5.41	5.50	5.33	5.352	0.109
5	5.47	5.24	5.27	5.25	5.35	5.316	0.096
6	5.33	5.30	5.30	5.24	5.38	5.310	0.051
7	5.46	5.24	5.22	5.39	5.30	5.322	0.102
8	5.30	5.36	5.39	5.25	5.24	5.308	0.066

Table 30. Computational Time (Seconds) for *SP 15* Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	513.08	499.31	439.06	439.34	433.62	464.882	38.095
2	502.15	436.10	440.50	437.20	437.05	450.600	28.865
3	440.94	438.57	443.52	435.92	441.85	440.160	2.969
4	441.28	440.55	443.14	439.17	440.34	440.896	1.466
5	438.20	443.68	444.59	467.30	441.99	447.152	11.525

Table 30. (Continued)							
6	481.53	457.00	482.22	449.19	474.27	468.842	14.965
7	429.12	467.60	473.71	442.58	440.56	450.714	19.036
8	461.51	474.56	439.29	451.38	455.97	456.542	12.980

Table 31. Computational Time (Seconds) for *LNM 15* Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std. Dev
1	47.31	45.45	45.07	45.10	45.71	45.728	0.923
2	45.00	45.48	45.29	45.73	46.86	45.672	0.716
3	45.85	46.17	46.26	46.81	53.96	47.810	3.455
4	45.43	45.28	44.62	44.89	44.70	44.984	0.357
5	45.00	44.93	44.64	46.20	45.40	45.234	0.604
6	45.96	45.48	45.10	44.71	45.37	45.324	0.463
7	45.87	45.32	45.46	45.40	44.81	45.372	0.379
8	45.17	44.67	46.48	45.39	44.95	45.332	0.695

Table 32. Computational Time (Seconds) for *SP 30* Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	624.93	626.75	624.97	627.08	705.14	641.774	35.436
2	706.26	705.23	626.52	703.10	624.31	673.084	43.538
3	626.75	624.77	628.16	628.93	710.84	643.890	37.460
4	628.098	631.011	625.77	628.68	634.08	629.528	3.155
5	627.43	627.65	712.59	710.92	674.91	670.700	42.177
6	632.24	633.35	625.87	714.28	627.29	646.606	37.964
7	627.49	629.84	628.16	632.16	635.71	630.672	3.344
8	630.00	708.11	626.7	710.03	626.28	660.224	44.619

Table 33. Computational Time (Seconds) for *LNM 30* Departments - *Sorting*

PN	R1	R2	R3	R4	R5	Average	Std Dev
1	45.75	47.31	49.28	48.92	47.26	47.704	1.426
2	47.81	47.54	48.96	47.53	47.87	47.942	0.590
3	47.87	48.40	48.21	47.48	47.42	47.876	0.433
4	48.46	48.50	48.71	48.48	48.25	48.480	0.163
5	48.42	48.14	49.39	49.57	48.29	48.762	0.666
6	48.39	48.23	47.75	45.54	49.01	48.345	0.520
7	46.21	45.46	45.53	45.89	45.45	45.708	0.333
8	45.64	45.20	46.10	45.65	46.26	45.770	0.420

5.7 *Sorting* vs no-*sorting* costs comparison

The comparison of costs for *sorting (S)* and *no-sorting* (NS) variants *is* provided in tables 34-36. The formula computed in the percentage differences is: *100*(Cost no-sorting – Cost sorting)/Cost no-sorting*. Then positive percentages mean that *sorting* decreased cost while negative percentages mean *sorting* increased it.

The comparison of *sorting* and *no-sorting* for the *PSP* show the considerable improvement achieved by *sorting* the 400,000 generated permutations and filtering the best 50,000 in each year instead of just generating 85,000 permutations and using them for all years. *Sorting* reduces costs by up to 1.34% if compared to *no-sorting*. However, *sorting* gave costs higher than *no-sorting* in 3 instances. In the remaining 21 instances, *sorting* outperformed *no-sorting*. All tables have 8 rows. Each row corresponds to each one of the 8 problems tested under the number of departments studied.

The comparison of *sorting* and *no-sorting* for the *LNM* shows the considerable difference achieved by *sorting* the 85,000 generated permutations and filtering the best 2,000 in each year. *Sorting* reduces the cost by up to 2.12 % if compared to *no-sorting*. *Sorting* also gives costs much higher than *no-sorting* for 3 instances but in the rest of the 21 instances *sorting* outperforms *no-sorting* in terms of cost.

5.8 *Sorting* vs *no-sorting* computational time comparison

The comparison of computational time for *sorting* and *no-sorting* is also in Tables 34-36. The formula computed in the percentage differences is: *100*(Computational Times no-sorting – computational times sorting)/computational times no-sorting*. Then positive percentages mean that *sorting* decreased time while negative percentages mean *sorting* increased it.

The comparison of *SP sorting* and PSP *no-sorting* shows the increase in time if sorting 400,000 permutations by the material handling cost of each year and filtering the best 50,000 instead of just generating 85,000 permutations and using them for all years (*PSP no-sorting*). The main reasons for the time increase are: (1) the number of permutations considered (400,000 *sorting*, 85,000 *no-sorting*) and (2) the lack of parallelization in the *sorting* variant. It translates into a time increase of up to 1175%. Computational times for 6-department problems in *sorting* were effective in comparison to the 6 departments problem in *no–sorting* while *no-sorting* gave effective computational times for the 15 and 30 department problems.

If comparing the computational time of *LNM sorting* and *LNM no-sorting*, *sorting* gave effective computational time in comparison to *no-sorting* with a 53% of maximum decrease in times. Both the *sorting* and *no-sorting* variants in AMPL use the node-arc formulation and therefore there should be no reason for the vast difference in computational times. It seems the only difference is that AMPL was able to execute the network simplex method faster on the network with the permutations for the sorted case than in the one for the no sorted case.

Table 34. Difference in Cost and Computational Time for *PSP* and *LNM - Sorting* (S) vs. *No- Sorting* (NS) - 6 Departments

	Cost					
Problem Number	*PSP-S*	*PSP-NS*	% Diff	*LNM - S*	*LNM-NS*	% Diff
1	106419	106419	0	106419	106419	0
2	105341	105341	0	105341	105341	0
3	102989	102989	0	102989	102989	0
4	106399	106399	0	106399	106399	0
5	105628	105628	0	105628	105628	0
6	103985	103985	0	103985	103985	0
7	106439	106439	0	106439	106439	0

Table 34. (Continued)						
8	103771	103771	0	103771	103771	0
	Time (Seconds)					
Problem Number	*SP-S*	*PSP-NS*	% Diff	*LNM-S*	*LNM-NS*	% Diff
1	0.12	1.03	88	5.37	5.05	-6
2	0.12	0.13	6	5.37	11.14	52
3	0.12	0.15	19	5.57	11.14	50
4	0.12	0.92	87	5.35	11.19	52
5	0.12	2.44	95	5.32	11.11	52
6	0.12	0.19	36	5.31	11.17	52
7	0.12	0.2	39	5.32	11.24	53
8	0.12	0.15	19	5.31	11.47	54

Table 35. Difference in Cost and Computational Time for *PSP* and *LNM* - *Sorting* (S) vs. *No-Sorting* (NS) - 15 Departments

	Cost					
Problem Number	*PSP-S*	*PSP-NS*	% Diff	*LNM -S*	*LNM-NS*	% Diff
1	498273	501547	0.65	501547	511087	1.9
2	499297	502998	0.74	502998	512633	1.9
3	500066	506883	**1.34**	507031	515212	1.59
4	498717	501035	0.46	501035	508173	1.4
5	499414	500312	0.18	500312	511144	**2.12**
6	499353	502486	0.62	502486	511498	1.76
7	501169	504100	0.58	504100	514077	1.94
8	505761	507402	0.32	507442	517778	2
	Time (Seconds)					
Problem Number	*SP-S*	*PSP-NS*	% Diff	*LNM-S*	*LNM-NS*	% Diff
1	464.9	36.5	-1175	45.73	64.65	29
2	450.6	97.4	-366	45.67	61.14	25
3	440.2	396.8	-10.94	47.81	62.08	22.99
4	440.9	396.1	-11.32	44.984	60.31	25.42
5	447.1	396.4	-12.79	45.234	60.43	25.1
6	468.8	396.1	-18.38	45.324	60.89	25.56
7	450.7	395.8	-13.89	45.372	60.29	24.74
8	456.5	384.05	-18.88	45.332	59.92	24.34

Table 36. Difference in Cost and Computational Time for *PSP* and *LNM* - *Sorting* (S) vs. *No* - *Sorting* (NS) - 30 Departments

	Cost					
Problem Number	*PSP-S*	*PSP-NS*	% Diff	*LNM-S*	*LNM-NS*	% Diff
1	603233	603165	-0.01	624464	609371	-2.48
2	603332	600408	-0.49	606909	606746	-0.03
3	605783	607720	0.32	612155	611773	-0.06
4	597999	601261	0.54	601721	607288	0.92
5	587107	590454	0.57	590548	599579	1.51
6	594994	597884	0.48	598842	603318	0.74
7	597179	595900	-0.21	599936	601822	0.31
8	601663	603972	0.38	603848	614843	1.79
	Time (Seconds)					
Problem Number	*SP-S*	*PSP-NS*	% Diff	*LNM-S*	*LNM-NS*	% Diff
1	641.77	457.01	-40.43	47.7	64.34	25.86
2	673.08	457.37	-47.16	47.94	71.9	33.32
3	643.89	456.4	-41.08	47.88	60.76	21.21
4	629.53	106.16	-493	48.48	60.03	19.23
5	670.7	107.75	-522.46	48.76	60.06	18.81
6	646.61	107.86	-499.49	48.35	77.62	37.71
7	630.67	111.2	-467.15	45.71	60.12	23.97
8	660.22	107.12	-516.34	45.77	76.85	40.45

Appendixes J, K and L present the number of network simplex iterations AMPL took to solve the *LNM* for all problems under the *sorting* and *no-sorting* variants studied. A series of paired t-tests with null hypothesis Ho: mean for the number of simplex iterations under *sorting* = mean for the number of iterations under *no-sorting* vs. Ha : mean for the number of iterations under *sorting* < mean for the number of iterations under *no sorting* show that for the problems with 6 departments the null hypothesis is rejected with a p-value of 2.99E-10 and for the problems with 15 departments the null hypothesis is rejected with a p-value of 1.92E-12.However, for the 30-departments case the hypothesis couldn't be rejected (p-value 0.43 vs. alpha 0.05). A paired t-test considering all the sample of 120

values (problems with 6, 15, and 30 departments, 8 in each case) lead to reject the null hypothesis with a p-value of 0.004. These tests permit to conclude that for problems of size 6 and 15 the number of simplex iterations in the *sorting* variant was statistically significantly less that for the *no-sorting* variant.

5.9 Cost and computational time differences between *PSP* and *LNM*

Tables 37 - 39 summarize the cost and computational time differences for the comparison between *PSP* and *LNM Model* under the two variants studied, *sorting (S)* and *no-sorting (NS)*. The formulas used to compute the cost differences are: *100*(cost PSP_NS – cost LNM_NS)/cost PSP_NS* and *100*(cost SP_S – cost LNM_S)/cost SP_S.* Then negative percentages in the first one mean that *PSP_NS* has lower cost than *LNM_NS*. Similarly, negative percentages in the second one mean that *SP_S* has lower cost than *LNM_S.*

However, the formulas used to compute the time differences are: *100*(time LNM_NS – time PSP_NS)/time LNM_NS* and *100*(time LNM_NS – time SP_NS)/time LNM_NS*. Then negative percentages in those formulas mean that *LNM* has lower computational time than *PSP* or *SP*. The comparison for the times between *SP_S* and *LNM_S* is not entirely fair since the times for *SP_S* include the serial time spent on generating and sorting the permutations which is not included in *LNM_S*. The reader can see a possibly fairer comparison of times between *SP_S* and *LNM_S* by looking at Appendix P where the differences in times to perform just the variant of the Dijkstra's algorithm are reported.

Table 37. Cost and Computational Time Differences between *PSP* and *LNM* - 6 Departments

	Cost *PSP*_NS vs *LNM_ NS*	Time *LNM_ NS* vs *PSP*_NS	Cost *SP*_S vs *LNM _S*	Time *LNM_ NS* vs *PSP*_NS
D_PN	% Diff	% Diff	% Diff	% Diff
6_1	0	79.64	0.00	97.73
6_2	0	98.86	0.00	97.73
6_3	0	98.70	0.00	97.81
6_4	0	91.74	0.00	97.72
6_5	0	78.03	0.73	97.71
6_6	0	98.30	0.00	97.70
6_7	0	98.20	0.00	97.71
6_8	0	98.73	0.00	97.70

Table 38. Cost and Computational Time Differences between *PSP* and *LNM* - 15 Departments

	Cost *PSP*_NS vs *LNM_ NS*	Time *LNM_ NS* vs *PSP*_NS	Cost *SP_S* vs *LNM _S*	Time *LNM_ NS* vs *PSP*_NS
D_PN	% Diff	% Diff	% Diff	% Diff
15_1	-1.90	43.60	-0.66	-922.33
15_2	-1.92	-59.32	-0.75	-886.60
15_3	-1.64	-539.11	-1.37	-820.64
15_4	-1.42	-556.68	-0.46	-880.10
15_5	**-2.17**	-555.97	-0.18	-888.53
15_6	-1.79	-550.49	-0.62	-934.42
15_7	-1.98	-556.45	-0.58	-893.37
15_8	-2.04	-540.95	-0.33	-907.10

Table 39. Cost and Computational Time Difference Between *PSP* and *LNM* - 30 Departments

	Cost *PSP*_NS vs *LNM_ NS*	Time *LNM_ NS* vs *PSP*_NS	Cost *SP_S* vs *LNM _S*	Time *LNM_ NS* vs *PSP*_NS
D_PN	% Diff	% Diff	% Diff	% Diff
30_1	-1.03	-610.3	**-3.40**	-1245.32
30_2	-1.06	-536.1	-0.59	-1303.95
30_3	-0.67	-651.1	-1.04	-1244.91
30_4	-1.00	-76.9	-0.62	-1198.51
30_5	**-1.55**	-79.4	-0.58	-1275.46
30_6	-0.91	-39.0	-0.64	-1237.47

Table 39. (Continued)				
30_7	-0.99	-85.0	-0.46	-1279.78
30_8	-1.80	-39.4	-0.36	-1342.47

5.10 DFLP costs and computational times comparisons between proposed methods and previous works

Tables 40-42 summarize the average costs and computational times for all the problems under the *PSP* and *LNM* methods and the variants (*sorting*, *no-sorting*) studied and the percentage cost difference vs. the *best known* cost from the literature. The computational time comparison vs. other authors couldn't be done since the information is not available. This comparison is also not meaningful since the computational settings used differ. Abbreviations used in Tables 40-42 are: number of departments and problem number (D_PN) and *best known* cost (BKC). Table 43 presents the average costs obtained by Conway & Venkataramanan (1994) with their genetic algorithm named CONGA, Balakrishnan & Cheng (2000) with their nested loop genetic algorithm named NLGA, Baykosaglu & Gindy (2001) with their simulated annealing (SA) method, Balakrishnan et al. (2003) with their hybrid dynamic programming and genetic algorithm approach named GADP, and McKendall & Shang (2006) with their hybrid ant systems methods named HASI, HASII, and HASIII.

Table 40. Cost, Computational Time and Percentage (%) of Cost Difference *PSP* and *LNM – Sorting*(S) and *No-Sorting*(NS) vs. *Best Known* Cost (BKC) - 6 Departments

	Parallel Shortest Path – no-sorting			*Linear Network Model – no-sorting*			
D_PN	Cost	Time (s)	% Cost vs. BKC	Cost	Time (s)	% Cost vs. BKC	BKC
6_1	106419	1.03	0	106419	5.05	0	106419
6_2	105341	0.13	0.48	105341	11.14	0.48	104834
6_3	102989	0.15	-1.28	102989	11.14	-1.28	104320
6_4	106399	0.92	0	106399	11.19	0	106399
6_5	105628	2.44	0.34	105628	11.11	0.34	105268
6_6	103985	0.19	0	103985	11.17	0	103985
6_7	106439	0.2	0	106439	11.24	0	106439
6_8	103771	0.15	0	103771	11.47	0	103771
	Shortest Path - sorting			*Linear Network Model - sorting*			
D_PN	Cost	Time (s)	% Cost vs. BKC	Cost	Time (s)	% vs. BKC	BKC
6_1	106419	0.12	0	106419	5.37	0	106419
6_2	105341	0.12	0.48	105341	5.37	0.48	104834
6_3	102989	0.12	-1.28	102989	5.57	-1.28	104320
6_4	106399	0.12	0	106399	5.35	0	106399
6_5	106399	0.12	1.07	105628	5.32	0.34	105268
6_6	103985	0.12	0	103985	5.31	0	103985
6_7	106439	0.12	0	106439	5.32	0	106439
6_8	103771	0.12	0	103771	5.31	0	103771

Table 41. Cost, Computational Time and Percentage (%) of Cost Difference *PSP* and *LNM – Sorting*(S) and *No-Sorting*(NS) vs. *Best Known* Cost (BKC) - 15 Departments

	Parallel Shortest Path – no-sorting			*Linear Network Model – no-sorting*			
D_PN	Cost	Time (s)	% Cost vs. BKC	Cost	Time (s)	% Cost vs. BKC	BKC
15_1	501547	36.46	4.39	511087	64.65	6.38	480453
15_2	502998	97.41	3.76	512633	61.14	5.75	484761
15_3	506883	396.77	3.71	515212	62.08	5.41	488748
15_4	501035	396.06	3.42	508173	60.31	4.9	484446
15_5	500312	396.43	2.58	511144	60.43	4.8	487722
15_6	502486	396.06	3.25	511498	60.89	5.1	486685
15_7	504100	395.76	3.54	514077	60.29	5.59	486853
15_8	507402	384.05	3.34	517778	59.92	5.45	491016
	Shortest Path - sorting			*Linear Network Model - sorting*			
D_PN	Cost	Time (s)	%Cost vs. BKC	Cost	Time (s)	% vs. BKC	BKC
15_1	498273	464.88	3.71	501547	45.73	4.39	480453
15_2	499297	450.6	3	502998	45.67	3.76	484761
15_3	500066	440.16	2.32	507031	47.81	3.74	488748
15_4	498717	440.9	2.95	501035	44.98	3.42	484446
15_5	499414	447.15	2.4	500312	45.23	2.58	487722
15_6	499353	468.84	2.6	502486	45.32	3.25	486685
15_7	501169	450.71	2.94	504100	45.37	3.54	486853
15_8	505761	456.54	3	507442	45.332	3.35	491016

Table 42. Cost, Computational Time and Percentage (%) of Cost Difference *PSP* and *LNM – Sorting*(S) and *No-Sorting*(NS) vs. *Best Known* Cost (BKC) - 30 Departments

	Parallel Shortest Path – no-sorting			*Linear Network Model – no- sorting*			
D_PN	Cost	Time (s)	% Cost vs. BKC	Cost	Time (s)	% Cost vs. BKC	BKC
30_1	603165	457.01	4.56	609371	64.34	5.63	576886
30_2	600408	457.37	5.27	606746	71.9	6.38	570349
30_3	607720	456.4	5.5	611773	60.76	6.2	576053
30_4	601261	106.16	6.08	607288	60.03	7.15	566777
30_5	590454	107.75	5.75	599579	60.06	7.38	558353
30_6	597884	107.86	5.49	603318	77.62	6.44	566792
30_7	595900	111.2	5.07	601822	60.12	6.12	567131
30_8	603972	107.12	4.99	614843	76.86	6.88	575280
	Shortest Path - sorting			*Linear Network Model - sorting*			
D_PN	Cost	Time (s)	% Cost vs. BKC	Cost	Time (s)	% Cost vs. BKC	BKC
30_1	603233	641.77	4.57	624464	47.7	8.25	576886
30_2	603332	673.08	5.78	606909	47.94	6.41	570349
30_3	605783	643.89	5.16	612155	47.88	6.27	576053
30_4	597999	629.53	5.51	601721	48.48	6.17	566777
30_5	587107	670.7	5.15	590548	48.76	5.77	558353
30_6	594994	646.61	4.98	598842	48.34	5.65	566792
30_7	597179	630.67	5.3	599936	45.71	5.78	567131
30_8	601663	660.22	4.59	603348	45.77	4.97	575280

Table 43. DFLP Average Costs presented by Other Authors for 6-15-30 Departments

Dep.	Problem	CONGA (1994)	NLGA (2000)	SA (2001)	GADP (2003)
6	1	108976	106419	107249	106419
6	2	105170	104834	105710	104834
6	3	104520	104320	104800	104320
6	4	106719	106515	106515	106515
6	5	105628	105268	106282	105628
6	6	105606	104053	103985	104053
6	7	106439	106978	106447	106439
6	8	104485	103771	103771	103771
15	1	504759	511854	501447	484090
15	2	514718	507694	506236	485352
15	3	516063	518461	512886	489898
15	4	508532	514242	504956	484625
15	5	515599	512834	509636	489885
15	6	509384	513763	508215	488640
15	7	512508	512722	508848	489378
15	8	514839	521116	512320	500779
30	1	632737	611794	604408	578689
30	2	647585	611873	604370	572232
30	3	642295	611664	603867	578527
30	4	634626	611766	596901	572057
30	5	639693	604564	591988	559777
30	6	637620	606010	599862	566792
30	7	640482	607134	600670	567873
30	8	635776	620183	610474	575720

Table 43. (Continued)

Dep.	Problem	HAS I (2006)	HAS II (2006)	HAS III (2006)
6	1	106419	106419	106419
6	2	104834	104834	104834
6	3	104320	104320	104320
6	4	106399	106399	106399
6	5	105628	105628	105628
6	6	103985	103985	103985
6	7	106439	106439	106439
6	8	103771	103771	103771
15	1	481511	481395	480453
15	2	484761	484761	484879
15	3	490899	488748	490398
15	4	485561	485658	484446
15	5	489012	487722	489206
15	6	487417	486685	486965
15	7	486853	486853	486853
15	8	493963	492074	491016
30	1	578854	576886	580240
30	2	570349	571528	570349
30	3	578152	576053	578176
30	4	569694	572005	566777
30	5	560433	558353	558353
30	6	569725	570567	566792
30	7	570899	567190	567131
30	8	576980	575998	575280

Figure 17 presents the costs for *PSP* and *LNM* for both the *sorting* and *no-sorting* variants studied. Figure 18 presents the costs for *PSP* sorted and *LNM* sorted vs. the ones from several previous authors, CONGA (1994), HAS III (2006) and SA (2001). Figure 19 presents the cost for *PSP* and *LNM* under *sorting* and *no-sorting* variants if compared to the *best known* cost.

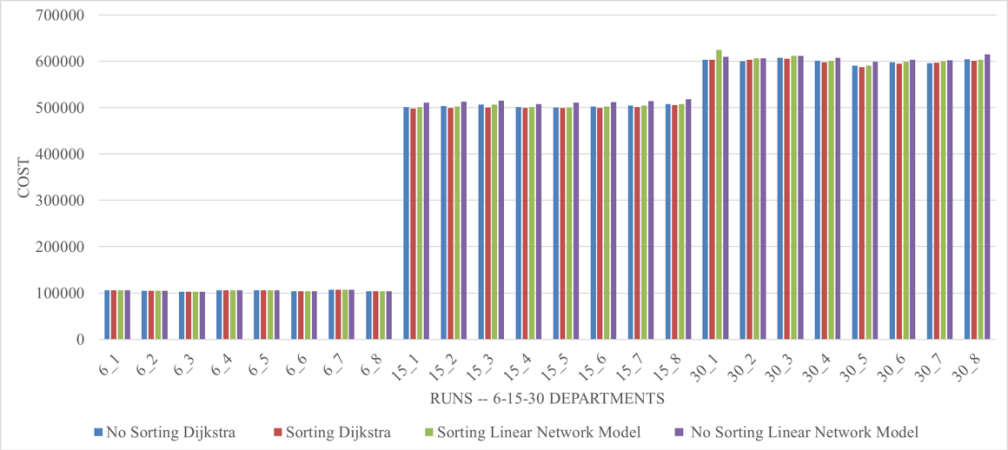


Figure 17. Costs for *PSP* and LNM under the *sorting* and no-*sorting* variants studied

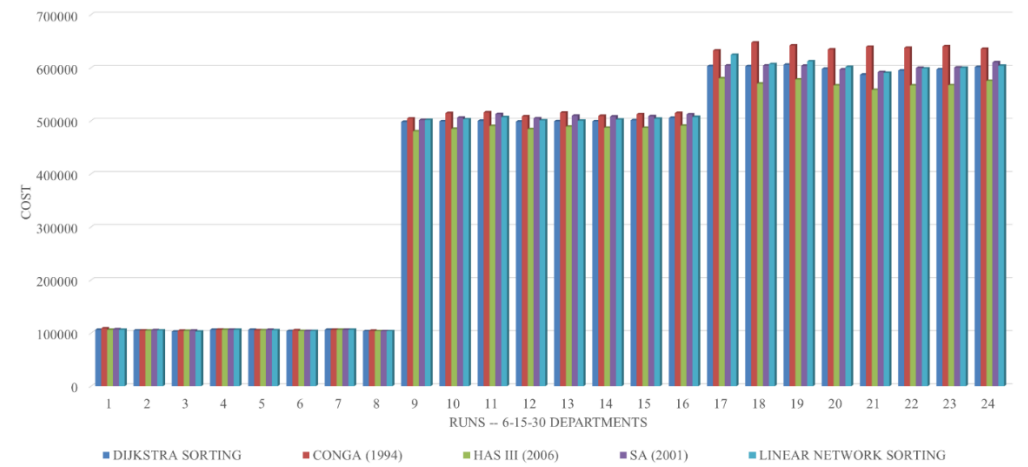


Figure 18. Costs *PSP sorting* and *LNM sorting* vs. previous authors

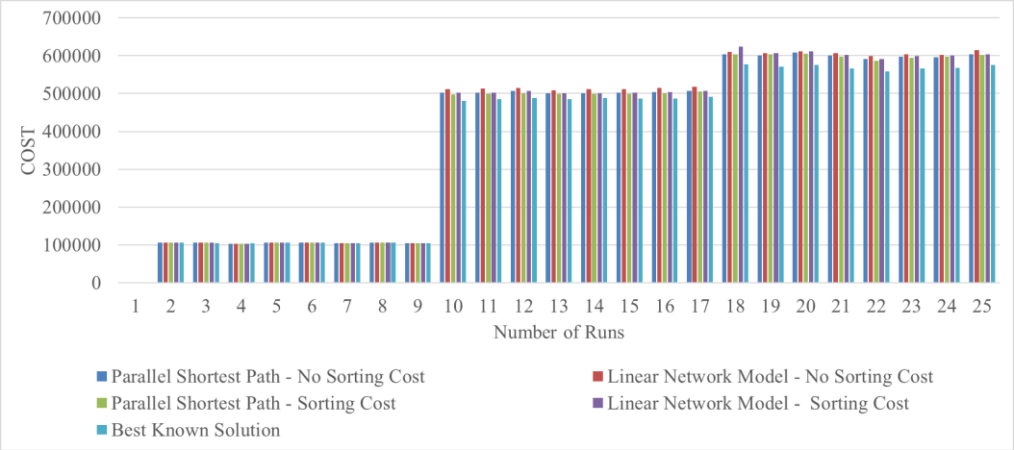


Figure 19. Cost for *PSP* and *LNM* (*sorting* and *no-sorting*) vs. *best known* cost

Appendix I has samples of output for best solutions found for the *PSP and LNM* algorithms for the *no-sorting* variant. The samples for the *sorting* variant were omitted to keep the document shorter.

5.11 Analysis of the results for the DFLP

The variation in cost and computational times over the 5 runs in each problem (see Tables 7-33) does not reveal any outliers of practical significance. The number of permutations used in the *LNM* is 2000. This low number is due to the memory issues experienced when AMPL attempted to solve larger problems. However, the results achieved are incredibly flawless. The comparison of costs for the *LNM* to the ones in the *PSP* and to the *best known* cost supports this statement.

Tables 37-39 show that the maximum increase in cost for *LNM* (*no–sorting*) vs. *PSP* (*no-sorting*) is only 2.17%. Also, the computational time comparison shows that *LNM* (*no–sorting*) results 651.13% faster than *PSP* (*no- sorting*). On the other hand, Tables 37-39 also show that the maximum increase in cost for *LNM* (*sorting*) if compared to SP *(sorting*) is only 3.40%. Also, the computational time comparison shows that r *LNM* (*sorting*) is up to 1342.47% faster than *SP (sorting).*

Tables 40-42 show that the maximum difference in cost for *PSP* (*no-sorting*) vs. *best known cost* is 6.08% and for *LNM* (*no-sorting*) vs. *best known cost* is 7.38%. Tables 40-42 also show that the maximum difference in cost for *PSP* (*sorting*) vs. *best known cost* is 5.78% and for *LNM* (*sorting*) vs. *best known cost* is 6.41% (ignoring the 8.25% in the first problem of size 30). Thus, the idea of filtering the permutations in each year performed well and improved the overall results.

A strength found in the *LNM* is its ability to produce similar accurate results with small number of permutations (i.e. layouts in the network per each period). The *LNM* gave also smaller computational times more often than the *PSP*. However, the comparison is not straightforward because of the different number of permutations (i.e. layouts) included under each case. Nevertheless, it is predicted that under a low number of permutations both methods are computationally efficient. The maximum running time for *PSP* (*no-sorting*) was 457.37 seconds (about 7 minutes) and for *LNM* (*no-sorting*) was 77.62 seconds. The maximum running time for *SP* (*sorting*) was 673.08 seconds and for *LNM* (*sorting*) was 48.76 seconds.

Table 44 shows the performance of the *PSP* and *LNM* under the best variant (i.e. the sorted variant) in comparison to 7 algorithms from previous authors studying the DFLP for the 24 problems considered. *PSP* achieved a better cost on 67 cases out of the total 24*7=168 compared. *LNM* gave a better cost than the *best known* cost on 62 cases. Both *LNM* and *PSP* equaled the *best known* cost in 5 occasions and surpassed it 1 case. *LNM* and *PSP* ended costlier 16 and 24 cases, respectively.

Table 44. Performance of *PSP* and *LNM* vs. Previous Authors Solutions

Algorithms from previous authors	Number of comparisons where Sorted *PSP* costs are better	Number of comparisons where Sorted *LNM* costs are better
CONGA (1994)	21	21
NLGA (2000)	20	18
SA (2001)	20	17
GADP (2003)	3	3
HAS I (2006)	1	1
HAS II (2006)	1	1
HAS III (2006)	1	1

5.12 SDFLP results

Table 45 presents the average objective function values resulting from solving the model proposed to solve the SDFLP presented in Chapter 2 Section 5. The periods are assumed to be 5 years. The problems generated have 3 scenarios (high, medium, low) for the budget available for relocations A_{st}, the flows between departments and the costs of relocating each department (UC's). The scenarios have probabilities 0.3, 0.5 and 0.2. A total of 720 layouts (i.e. 6!) are included in each year. Costs of relocating each department (UC's) follow a probability distribution because of variations in charges due to labor, fuel and special equipment's used during relocations.

Experimentation was done under three cases. Case 1 corresponds to very high magnitudes for the values of the A_{st} parameters; still decreasing the numbers over the 3 scenarios considered. In case 1, the model was run 4 times in each of the 4 six departments problems generated (P1, P2, P3, P4). In cases 2 and 3 the model was run also 4 times but only for 1 of the six departments problem generated (P1). Cases 2 and 3 correspond to medium and low magnitudes for the values for A_{st}, still decreasing the numbers over the

3 scenarios considered. The budgeted money for relocations, A_{st}, includes the loss in production incurred during the time the departments are under relocation.

The terms in the SDFLP objective function and their signs introduced in Chapter 2 are pasted below again.

$$min\ z_{stoc} = \sum_{s=1}^{S'} \sum_{i_t}^{L_t} \sum_{K_{t+1}}^{L_{t+1}} p_s C_{si_tK_{t+1}} x_{i_tK_{t+1}} - \sum_{s=1}^{S'} p_s y_{s_{(N-1)}} (1+r)^{-(N-1)} \qquad (11)$$

However, in table 45, the sign for z_{stoc} is displayed as minus the value computed in equation (11). This means a negative number for z_{stoc} represents the case in which besides incurring in MHC and RC, money should be borrowed to relocate the departments (negative values for the *y* variables) and thus z_{stoc} ends being a cost. A positive number for z_{stoc} indicates that the total amount unused from the allocated budget (positive values for the *y* variables) exceeded the cost incurred in MHC and RC.

These preliminary experiments indicate that the model is sensitive to A_{S_t}, the available or allocated budget for the relocations occurring between period *t* and *t+1* under scenario *s*. Future work will consider calibrating the model parameters to resemble a company setting and to assess the value of the stochastic solution from this SDFLP model vs. the one from solving a DFLP.

Table 45. Results from Solving the SDFLP - 6 Departments

Case	Problem	$-z_{stoc}$ (see equation 11)
High magnitudes for the budget for relocations , A_{st}	P1	195,729
	P2	62,813
	P3	34,132
	P4	193,948
Medium magnitudes for the budget for relocations , A_{st}	P1	11,938
Low magnitude for the budget for relocations , A_{st}	P1	-62,000

6. PRACTICAL CASE STUDY

6.1 Micro Power facility layout problem

Micro Power Global, a private green energy technology company started in 2008 and partnered with Texas State University to develop and commercialize a cutting-edge solid-state semiconductor which in the Power Mode converts heat directly into electricity and in the cooling mode converts electricity directly into refrigeration (MicroPower Global; Novoa & Mai, 2013). Texas State University has provided technical expertise, facilities and equipment to Micro Power.

In 2012, MicroPower starts to relocate its operations to the new STAR park facility in San Marcos, TX in preparation for company's growth. Uncertainty and variability of product demands require a flexible facility layout that may involve shifting of some departments onto new locations inside the building at certain periods of time or a careful decision on the final position of each department if they are not going to be relocated.

The manufacturing process for the wafers produced by MicroPower is sequential and involves 5 production departments in this order: ingot growing, ingot slicing, wafer polishing, MBE, and metrology. MicroPower needs a facility layout to accommodate these departments. The STAR park facility has been divided into 6 equal rectangular locations labeled as shown in Figure 20.

Figure 20. Numerical labels for the locations in the Micro Power STAR Park facility

Table 46 shows the distances between the locations as labeled and depicted in Figure 20. A dummy department is added to have an equal number of departments and locations. Flows of material (in trips/year) between departments over the next 3 years, based on the product demands and material handling systems used, are collected through meetings with the managers and presented in Tables 47 and 48. MicroPower has provided also estimates for the departments' relocation costs and they are in Table 49. Given the information collected, this problem matches the definition and assumptions in a DFLP problem with deterministic but variable material flow.

Table 46. Distances (in Feet) between Locations at the Micro Power Facility

Location\Location	1	2	3	4	5	6
1	0	32	28	62	56	89
2	32	0	61	29	88	57
3	28	61	0	32	27	60
4	30	29	32	0	59	27
5	56	88	27	59	0	32
6	89	57	60	27	32	0

Table 47. Flows of Material between Micro Power Departments for Year 1 (in Trips/Year)

Departments	Ingot	Slicing	Polishing	MBE	Metrology	Dummy
Ingot growing	0	53000	0	0	5300	0
Slicing	0	0	53000	0	2650	0
Polishing	0	0	0	53000	2650	0
MBE	0	0	0	0	5300	0
Metrology	5300	2650	2650	5300	0	0
Dummy	0	0	0	0	0	0

Table 48. Flows of Material between Micro Power Departments for Year 2 and Year 3 (in Trips/Year)

Departments	Ingot	Slicing	Polishing	MBE	Metrology	Dummy
Ingot growing	0	363,000	0	0	36300	0
Slicing	0	0	363,000	0	18,150	0
Polishing	0	0	0	363,000	18,150	0
MBE	0	0	0	0	36,300	0
Metrology	36,300	18,150	18,150	36,300	0	0
Dummy	0	0	0	0	0	0

Table 49. Departments Relocation Costs at Micro Power

Department	Fixed relocation Cost
Ingot growing	200,000
Slicing	100,000
Polishing	250,000
MBE	1,000,000
Metrology	200,000
Dummy	0

6.2. Results

The two methodologies described in this thesis (*PSP* and *LNM*) are tested to assess their practical suitability for solving the DFLP faced by MicroPower. Because of the relatively small size of this DFLP if modeled as a network problem, (6! = 720 layouts per period, a network with 720*3 + 2 = 2162 nodes and (T-1)*(n!*n!) + 2n! = 2*(720*720) +

2*720 = 1,038,240 arcs), the input to the *PSP* and *LNM* models was to include all 720 layouts or permutations for each year in the networks.

PSP and *LNM* methodologies gave the same result. The permutation **4-6-5-3-1-2** is the new layout suggested for the MicroPower facility by the two methodologies. It means department 4, MBE, must be in location 1, department 6, Dummy, must be in location 2, department 5, Metrology, must be in location 3, department 3, Polishing, must be in location 4, department 1, Ingot must be in location 5 and department 2, Slicing, must be in location 6. The total cost of such solution is $84,982,000. It results from summing the material handling costs incurred over the 3 years. No relocation costs are added since no department is relocated. Figure 21 presents the final facility layout suggested for MicroPower and Figure 22 presents the mirror image of it. The mirror image shows flows between departments from left to right and consequently it agrees more with the flow concepts suggested in facility layout books. The *LNM* implemented in AMPL takes about 2 seconds to solve this problem and performs 8,576 network simplex iterations. The serial *SP* takes about 0.06 seconds in the CAPI machine.

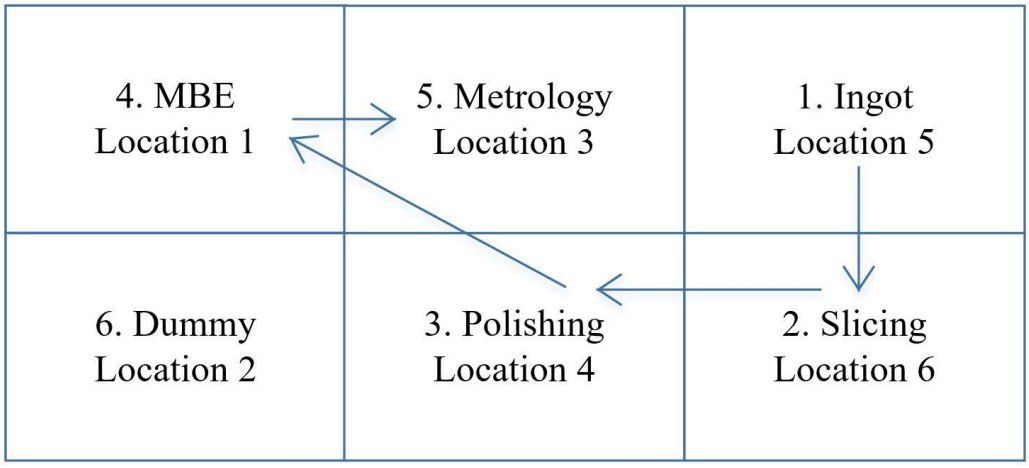


Figure 21. New layout suggested for Micro Power after solving with *PSP* and *LNM* methods

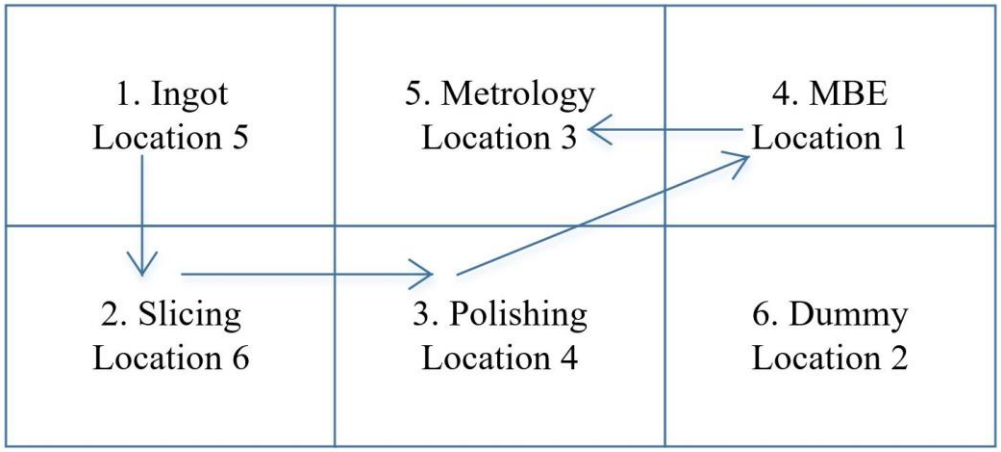


Figure 22. Mirror image for new layout suggested for Micro Power after solving with *PSP* and *LNM* methods

It is nice to see that without imposing any additional constraint the ingot growing and ingot slicing departments ended as neighbors. In previous works (Novoa & Mai, 2013) managers required to allocate both departments in a single location since in reality it is highly desirable to have these departments closer. The total cost found in this study cannot be directly compared to the one in Novoa & Mai (2013) because in this thesis 5 different departments are allocated to 5 locations and in Novoa & Mai (2013) after adding to the non-linear model the constraint that ingot growing and ingot slicing must occupy the same location the solution has 5 departments allocated to 4 locations.

It is also nice to see another couple of results that the managers desired: (1) MBE (department 4) is closer to polishing (department 3) and metrology (department 5), and (2) metrology (department 5) is closer to ingot (department 1), slicing (department 2), polishing (department 3) and MBE (department 4).

A layout for the departments was randomly generated (permutation 5-1-3-6-2-4) and the cost of keeping such layout over the 3-years was computed. This cost is equal to 162,107,300 and it is significantly higher if compared to the optimal one of $84,982,000

(permutation 4-6-5-3-1-2). The random layout doesn't have ingot close to slicing, MBE is relatively close to polishing but not close to metrology, and metrology is located closer to polishing and ingot but not closer to MBE and slicing. In this way, the random layout can be considered as a very poor one. The difference in costs should be a fair assessment of the maximum amount of money that can be saved over 3-years if using optimization techniques in facilities planning.

Also, the cost for one case in which 3 different and sub-optimal layouts are used for each year was computed. Such cost ended equal to 132,319,450. In this case, some layouts were not as poor as the random one mentioned in the previous paragraph but there were unnecessary relocation costs. For this particular problem, relocation costs are very high for the MBE department and therefore they should be avoided, if possible. Due to the small problem size and the rectangular shape of the facility, it is possible to allocate a department that has high flows with multiple departments in a central location and pair of departments that have high flows as neighbors. Such fact turns into the fact that the optimal solution ends to be a fixed layout instead of a different one for each year.

7. CONCLUSION

In this research, the dynamic facility layout problem (DFLP) and the stochastic and dynamic facility layout problem (SDFLP) have been studied. Two methodologies for solving the DFLP have been compared, *Parallel Shortest Path (PSP)* solved under a slightly modified version of the Dijkstra algorithm and a Linear Network Model (*LNM*) solved in AMPL. In the first phase of this study, solutions are obtained without filtering the permutations to include in the network. The efficiency of the proposed methodologies/algorithms is analyzed. This phase is called the *non-sorting variant*. Later, in the second phase the proposed methodologies are analyzed by filtering out the layouts with the best material handling cost in each year. The methodologies are contrasted in terms of cost of the solutions and computational time. This second phase is called the *sorting variant*. Both methodologies prove to be efficient and relatively simple to develop if compared to other heuristic algorithms previously published by other authors. They have also acceptable percentages of accuracy.

For the *non-sorting* variant, this thesis performed the experimental work on the *PSP* implementation developed by Kolla, (2015) by including a very large number of randomly selected layouts (85,000) to get the best quality results and perform comparisons to methodologies from previous authors. In this study, the *sorting variant* compared well to other previous works by generating 400,000 layouts and filtering the 50,000 with the best material handling cost in the *PSP* methodology and by generating 85,000 layouts and filtering the 2,000 ones with the best material handling cost in the *LNM* methodology.

This thesis proposed a new model for solving the SDFLP and demonstrated that the problem can be solved with a model that is an extension of the *LNM* used for the DFLP.

Problems with 6 departments and 5-time periods were solved with AMPL. Future research includes to extend the experimentation to see up to what problem sizes are possible to solve with the proposed methodology and to assess the value of the stochastic solution. SDFLP is a more realistic problem arising if considering that product demands (and consequently flows of material between departments) and relocation costs are not known with certainty but only through probability distributions.

This research proves to be the first one in solving models for both DFLP for 6,12,15 and 30 departments and SDFLP with budget constraints and 6 departments using a linear model and solving directly with an operations research software such as AMPL. It was rewarding to discover that AMPL was able to solve those problems using the network simplex algorithm. By using AMPL this work demonstrated that is possible to include up to 2000 layouts per period in the network. This number is a significant improvement if compared to the numbers reported for the DFLP in Balakrishnan & Jacobs (1992). These authors are the only ones researching previously on the Linear Network Model approach. They considered two settings, one with 50 layouts per period (small) and the other one with 100 layouts (large) per period.

The computational time reported by AMPL/CPLEX on solving the instances studied evidences that the software is efficient in solving these NP-Hard problems. A computer with more memory could be able to solve the DFLP and SDFLP problems with a larger number of layouts and to possibly get improvements in the costs.

The two methodologies for solving the DFLP can be used in industry. The *LNM* may be simpler to understand and perform to a supply chain/logistics practitioner than the *PSP*. *LNM* is relatively accurate and useful for those companies not in search of the very

optimal solution but just a good solution or a solution that is better to the current implemented one, and efficient in terms of computational time. However, *LNM* relies on running separately a C program to compute the some of the model input parameters (total costs equal to the sum of material handling cost and relocation costs) and to prepare the very large data file needed to run the model. On the other hand, the *PSP* is a single C program that can be used by a practitioner as a black box once he/she is trained on running the parallelized code on a parallel environment with OpenMP capabilities. Finally, the comparison of the methods studied in this thesis was not entirely straightforward since the *PSP* has been parallelized in the *non-sorting* variant and the *LNM* has been limited on the number of layouts to include because of computer memory constraints. Both methods were put in the best conditions they can work to identify their solution quality vs. other methods reported in the literature.

From a practical perspective, achieving cost reductions that considerably outperformed other methods in 67 and 62 cases by a percentage between 1-4 % is a result having a substantial value. This is because the 1-4% cost improvements for the problems studied (6,15 and 30 department's) will positively impact a company budget by freeing a significant amount of money that can be saved or re-allocated to cover other operational costs. From the research point of view, for this NP-hard problem authors have used various methodologies to achieve reduction in cost however for most of the problems no one knows the exact lowest cost but it is of high interest for the research community to know which algorithms are the best performers for the particular instances and which ones do it efficiently. Notoriously, we believe that most of the prior solution approaches have used more complex algorithms to explore the search space to find the minimum cost for the

instances studied. Hence, it is of great value for a company that is considering to implement *PSP* and *LNM* to setup their layout to have a not too complex algorithm available that has been contrasted to previous works and it produces good results in terms of costs.

This work helps supply chains to cut operational costs such as material handling and relocation cost. It helps to prioritize relocation of departments each year considering demand forecasts, new products, unexpected disasters, changes in machinery, geography or other issues. The places for applying the proposed methodologies are wide (warehouses, distribution centers, manufacturing plants, telecommunication and healthcare companies, and disaster relief supply chains among others). Future research will extend these models and research on methodologies to solve larger instances of the DFLP and SDFLP. The SDFLP is a facility layout problem where literature is scarce, especially efficient and good quality solution methods.

APPENDIX

APPENDIX A: Table 50: Summary of Literature Reviewed on DFLP

<table>
<tr><th colspan="4">DFLP</th></tr>
<tr><td colspan="2">Heuristics and Approximate Dynamic Programing</td><td colspan="2">Metaheuristics</td></tr>
<tr><td>Author</td><td>Approach</td><td>Author</td><td>Approach</td></tr>
<tr><td rowspan="2">Urban (1998)</td><td rowspan="2">GRASP and Initialized Multi Greedy Algorithms compared to DP</td><td>Conway and Venkataramanan (1994)</td><td rowspan="2">Evolutionary Algorithms - Genetic Algorithms (GA)</td></tr>
<tr><td>Conway and Venkataramanan (1994) Balakrishnan and Cheng (2000)</td></tr>
<tr><th colspan="4">DFLP</th></tr>
<tr><td colspan="2">Heuristics and Approximate Dynamic Programing</td><td colspan="2">Metaheuristics</td></tr>
<tr><td>Author</td><td>Approach</td><td>Author</td><td>Approach</td></tr>
<tr><td>Lacksonen and Enscore (1993)</td><td>Several exact methods (approximate for large instances) including DP</td><td>Kaku and Mazzola (1997)</td><td>Tabu Search (TS)</td></tr>
<tr><th colspan="4">DFLP</th></tr>
<tr><td colspan="2">Heuristics and Approximate Dynamic Programing</td><td colspan="2">Metaheuristics</td></tr>
<tr><td>Author</td><td>Approach</td><td>Author</td><td>Approach</td></tr>
<tr><td rowspan="2">Balakrishnan & Cheng (2009)</td><td>CRAFT, Urban heuristic, DP and backward pass algorithm</td><td>Baykasoglu and Gindy (2001) Moslemipour & Lee (2012)</td><td>Simulated Annealing (SA)</td></tr>
<tr><td></td><td>Baykasoglu et al. (2006)</td><td>Ant Systems</td></tr>
<tr><td colspan="4"></td></tr>
</table>

Table 50 (Continued)	**DFLP**		
Exact		Hybrid	
Author	Approach	Author	Approach
Rosenblatt (1986)	Dynamic Programming (DP)	Balakrishnan et al. (2003) Balakrishnan and Cheng (2006)	DP and GA

DFLP			
Exact		Hybrid	
Author	Approach	Author	Approach
Balakrishnan et al. (1992)	Linear programming to solve a network model using a specialized primal shortest-path simplex algorithm (LP)	McKendall and Shang (2006)	Ant Systems and Simulated Annealing

DFLP	
Exact	
Author	Approach
Urban (1998)	Exact Dynamic Programming (DP) small instances

APPENDIX B: Table 51: Summary of Literature Reviewed on SDFLP

SDFLP					
Heuristics		Metaheuristics		Exact	
Author	Approach	Author	Approach	Author	Approach
Benjaafar and Sheikhzadeh (1997)	CRAFT	Vitayasak et al. (2017)	Genetic Algorithm(GA) and Backtracking Search Algorithm (BSA)	Palekar et al. (1992)	Dynamic Programming (DP)
		Krishnan et al. (2008)	Generic Algorithm (GA)		
		Tayal et al. (2016) Tayal et al. (2018)	Simulated Annealing (SA) and Chaotic Simulated Annealing (CSA)		

APPENDIX C: Table 52: AMPL Model File for DFLP without Budget

```
set Layouts;
set Arcs within (Layouts cross Layouts);
param supply {Layouts} >= 0; # amounts available at Layouts
param demand {Layouts} >= 0; # amounts required at Layouts
check: sum {i in Layouts} supply[i] = sum {j in Layouts} demand[j];
param cost {Arcs} >= 0;
param capacity {Arcs} >= 0; # max packages that can be shipped
minimize Total_Cost;
node Balance {k in Layouts}: net_in = demand[k] - supply[k];
arc Ship {(i,j) in Arcs} >= 0, <= capacity[i,j],
from Balance[i], to Balance[j], obj Total_Cost cost[i,j];
```

APPENDIX D: Table 53: AMPL Data File for DFLP without Budget

```
set Layouts := 0 1 2 3 4 5 6 7 8 9…..721;
set Arcs :=
(   0   ,   1   )
(   0   ,   2   )
(   0   ,   3   )
.
.
(   719   ,   721   )
(   720   ,   721   );
param supply default 0 := 0 1;
param demand default 0 := 721 1;
param: cost capacity :=
0   1   0   800
0   2   0   800
0   3   0   800
.
.
720   721   14401    800;
```

APPENDIX E: **Table 54**: AMPL Model File for DFLP with Budget

```
set Layouts;
set Arcs within (Layouts cross Layouts);
param supply {Layouts} >= 0; # amounts available at Layouts
param demand {Layouts} >= 0; # amounts required at Layouts
check: sum {i in Layouts} supply[i] = sum {j in Layouts} demand[j];
param cost {Arcs} >= 0;
param capacity {Arcs} >= 0; # max units that can be shipped
param RC {Arcs} >= 0;
param budget >= 0;
minimize Total_Cost;
node Balance {k in Layouts}: net_in = demand[k] - supply[k];
arc Ship {(i,j) in Arcs} >= 0, <= capacity[i,j], from Balance[i], to Balance[j], obj
Total_Cost cost[i,j];
subject to BC:  sum {(i,k) in Arcs} RC[i,k]*Ship[i,k] <= budget;
```

APPENDIX F: Table 55: AMPL Data File for DFLP with Budget

```
set Layouts := 0 1 2 3 4 5 6 7 8……..721;
set Arcs :=
(   0   ,   1   )
(   0   ,   2   )
.
.
.
(   719   ,   721   )
(   720   ,   721   );
param supply default 0 := 0 1;
param demand default 0 := 721 1;
param budget := 3600;
param: cost capacity RC :=
0   1   0   800   900
0   2   0   800   900
.
.
.
719   721   14401   800   900
720   721   14401   800   900;
```

APPENDIX G: Table 56: AMPL Model file for SDFLP with Budget

```
set scenarios;
set years;
set yearminus1;
set Layouts;
set Layouts1;
set Layouts2;
set Layouts3;
set Layouts4;
set Layouts5;

param start in Layouts;
param end in Layouts, <> start;

set Arcs within (Layouts diff {end}) cross (Layouts diff {start});
set Arcs1 within (Layouts1 diff {end}) cross (Layouts2 diff {start});
set Arcs2 within (Layouts2 diff {end}) cross (Layouts3 diff {start});
set Arcs3 within (Layouts3 diff {end}) cross (Layouts4 diff {start});
set Arcs4 within (Layouts4 diff {end}) cross (Layouts5 diff {start});

param supply {Layouts} >= 0; # amounts of imaginary unit available at the
nodes or Layouts. Only dummy supply node sends a unit
param demand {Layouts} >= 0; # amounts of imaginary unit required by the
nodes or Layouts. Only destination node demands a unit
param numscen;
param probscen {scenarios};
param cost {Arcs, scenarios} >= 0; # Material handling cost C plus relocation
costs
param capacity {Arcs} >= 0; # nodes (layouts) have infinite capacity to transfer
the single "fictitious" unit passing in the network
param RC {Arcs, scenarios} >= 0; # Relocation cost from one layout at period t
to another layout at period t+1
param budget_avail {scenarios, years} >= 0; # Budget allocated to relocations
that varies by scenario and year; A in our model
param interest; # interest rate used as discount in the objective function

var Ship {(i,j) in Arcs} >= 0;   # X's in our model in the thesis
var Remaining{scenarios, years};    # Y in our model in the thesis
var total_avail{scenarios,years};   # B in our model in the thesis

minimize Total_Cost: sum {(i,j) in Arcs, s in scenarios}
probscen[s]*cost[i,j,s]*Ship[i,j] - sum{s in scenarios}
probscen[s]*Remaining[s,4]*(1 + interest)^(-4);
# 4 is to represent the previous to last year
```

Table 56. (Continued)

```
subject to Start: sum {(start, j) in Arcs} Ship[start,j] = 1;
subject to Balance {k in Layouts diff {start, end}}:
sum {(i,k) in Arcs} Ship[i,k] = sum {(k,j) in Arcs} Ship[k,j];

subject to Rem1to2 {s in scenarios}: sum {(i,k) in Arcs1 } RC[i,k,s]*Ship[i,k]
+ Remaining[s,1] = total_avail[s,1];
subject to Rem2to3 {s in scenarios}: sum {(i,k) in Arcs2 } RC[i,k,s]*Ship[i,k]
+ Remaining[s,2] = total_avail[s,2];
subject to Rem3to4 {s in scenarios}: sum {(i,k) in Arcs3 } RC[i,k,s]*Ship[i,k]
+ Remaining[s,3] = total_avail[s,3];
subject to Rem4to5 {s in scenarios}: sum {(i,k) in Arcs4 } RC[i,k,s]*Ship[i,k]
+ Remaining[s,4] = total_avail[s,4];

subject to Tie1to2 {s in scenarios}: budget_avail[s,1]  = total_avail[s,1];
subject to Tie2to3 {s in scenarios}: budget_avail[s,2] +
Remaining[s,1]*(1+interest) = total_avail[s,2];
subject to Tie3to4 {s in scenarios}: budget_avail[s,3] +
Remaining[s,2]*(1+interest) = total_avail[s,3];
subject to Tie4to5 {s in scenarios}: budget_avail[s,4] +
Remaining[s,3]*(1+interest) = total_avail[s,4];
```

APPENDIX H: Table 57: AMPL Data File for SDFLP with Budget

```
set Layouts := 0 1 2 3 4 5 6 7 8.............................721;
set Layouts1 := 1 2   3   4   5   6   7   8.................144;
set Layouts3 := 289  290   291   292   293   294   295   296.............432;
set Layouts4 := 433  434   435   436   437   438   439   440.............576;
set Layouts5 := 577  578   579   580   581   582   583   584.............720;
set years := 1 2 3 4 5;
set scenarios := 1 2 3;
set yearminus1 := 1 2 3 4;
set Arcs :=
(   0   ,   1   )
(   0   ,   2   )
.
.
.
(   719   ,   721   )
(   720   ,   721   );
set Arcs1 :=
(   1   ,   145   )
(   1   ,   146   )
.
.
.
set Arcs2 :=
(   145   ,   289   )
(   145   ,   290   )
.
.
.
(   288   ,   431   )
(   288   ,   432   );
set Arcs3 :=
(   289   ,   433   )
(   289   ,   434   )
.
.
.
(   432   ,   575   )
(   432   ,   576   );
set Arcs4 :=
(   433   ,   577   )
(   433   ,   578   )
.
.
```

Table 57. (Continued)

```
.
(   576   ,   719   )
(   576   ,   720   );
param start := 0;
param end := 721;
param supply default 0 := 0 1;
param demand default 0 := 721 1;
param numscen := 3;
param probscen := 1 0.3 2 0.5 3 0.2;
param budget_avail :
   1   2   3   4   5 :=
1 22200 23000 23500 24000 25000
2 35000 35500 36000 36500 37500
3 45000 45500 46000 46500 46500;
param interest := 0.04;
param   cost:  1     2   3  :=
0   1   12900    13700    14500
0   2   12900    13700    14500
.
.
.
719   721   12900    13700    14500
720   721   12900    13700    14500;
param capacity :=
0   1   800
0   2   800
.
.
.
719   721   800
720   721   800;
param RC:   1   2   3 :=
0   1   500   525   550
0   2   500   525   550
.
.
.
719   721   500   525   550
720   721   500   525   550;
```

APPENDIX I: Samples of best solutions found for the *PSP* algorithm developed by Chandra Kolla (2015) and the *LNM* coded in AMPL by Gowtham Balachandran - *no-sorting*

6 Department 1st problem 1st Run				
Chandra Parallel Algorithm		Balachandran Network Model - AMPL		
Optimal path:		Total_Cost = 106419		CPLEX 12.6.1.0: optimal solution; objective 106419
1 2 4 5 3 6 8				24972 network simplex iterations.
1 2 4 5 3 6 728		Runtime in seconds:		0 simplex iterations (0 in phase I)
1 2 4 5 3 6 1448		_ampl_user_time = 5.23438		Run time for solving the problem = 3.843750 seconds
1 2 4 5 3 6 2168				
1 2 4 5 3 6 2888		Ship :=		
		0 672 1	6 3 5 4 2 1	
Cost: 106419		672 1392 1	6 3 5 4 2 1	
		1392 2112 1	6 3 5 4 2 1	
real 0m0.866s		2112 2832 1	6 3 5 4 2 1	
user 0m1.350s		2832 3552 1	6 3 5 4 2 1	
sys 0m0.051s		3552 3601 1	6 3 5 4 2 1	

6 departments 1st problem 2nd Run				
Chandra Parallel Algorithm		Balachandran Network Model-AMPL		
Optimal path:		Total_Cost = 106419		CPLEX 12.6.1.0: optimal solution; objective 106419
1 2 4 5 3 6 8				24972 network simplex iterations.
1 2 4 5 3 6 728		Runtime in seconds:		0 simplex iterations (0 in phase I)
1 2 4 5 3 6 1448		_ampl_user_time = 4.98438		Run time for solving the problem = 3.625000 seconds
1 2 4 5 3 6 2168				
1 2 4 5 3 6 2888		Ship :=		
		0 672 1	6 3 5 4 2 1	
Cost: 106419		672 1392 1	6 3 5 4 2 1	
		1392 2112 1	6 3 5 4 2 1	
real 0m1.194s		2112 2832 1	6 3 5 4 2 1	
user 0m1.363s		2832 3552 1	6 3 5 4 2 1	
sys 0m0.057s		3552 3601 1	6 3 5 4 2 1	

6 Departments 1st problem 3rd Run				
Chandra Parallel Algorithm		Balachandran Network Model - AMPL		
Optimal path:		Total_Cost = 106419		CPLEX 12.6.1.0: optimal solution; objective 106419
1 2 4 5 3 6 8				24972 network simplex iterations.
1 2 4 5 3 6 728		Runtime in seconds:		0 simplex iterations (0 in phase I)
1 2 4 5 3 6 1448		_ampl_user_time = 4.95312		Run time for solving the problem = 3.609375 seconds
1 2 4 5 3 6 2168				
1 2 4 5 3 6 2888		Ship :=		
		0 672 1	6 3 5 4 2 1	
Cost: 106419		672 1392 1	6 3 5 4 2 1	
		1392 2112 1	6 3 5 4 2 1	
real 0m0.994s		2112 2832 1	6 3 5 4 2 1	
user 0m1.358s		2832 3552 1	6 3 5 4 2 1	
sys 0m0.051s		3552 3601 1	6 3 5 4 2 1	

6 Departments 1st Problem 4th Run										
Chandra Parallel Algorithm			Balachandran Network Model - AMPL							
Optimal path:			Total_Cost = 106419			CPLEX 12.6.1.0: optimal solution; objective 106419				
1 2 4 5 3 6 8						24972 network simplex iterations.				
1 2 4 5 3 6 728			Runtime in seconds:			0 simplex iterations (0 in phase I)				
1 2 4 5 3 6 1448			_ampl_user_time = 5.03125			Run time for solving the problem = 3.671875 seconds				
1 2 4 5 3 6 2168										
1 2 4 5 3 6 2888			Ship :=							
			0 672 1		6 3 5 4 2 1					
Cost: 106419			672 1392 1		6 3 5 4 2 1					
			1392 2112 1		6 3 5 4 2 1					
real 0m0.978s			2112 2832 1		6 3 5 4 2 1					
user 0m1.367s			2832 3552 1		6 3 5 4 2 1					
sys 0m0.045s			3552 3601 1		6 3 5 4 2 1					

6 Departments 1st Problem 5th Run									
Chandra Parallel Algorithm		Balachandran Network Model - AMPL							
Optimal path:		Total_Cost = 106419			CPLEX 12.6.1.0: optimal solution; objective 106419				
1 2 4 5 3 6 8					24972 network simplex iterations.				
1 2 4 5 3 6 728		Runtime in seconds:			0 simplex iterations (0 in phase I)				
1 2 4 5 3 6 1448		_ampl_user_time = 5.0625			Run time for solving the problem = 3.687500 seconds				
1 2 4 5 3 6 2168									
1 2 4 5 3 6 2888		Ship :=							
		0 672 1		6 3 5 4 2 1					
Cost: 106419		672 1392 1		6 3 5 4 2 1					
		1392 2112 1		6 3 5 4 2 1					
real 0m1.111s		2112 2832 1		6 3 5 4 2 1					
user 0m1.340s		2832 3552 1		6 3 5 4 2 1					
sys 0m0.056s		3552 3601 1		6 3 5 4 2 1					

12 Departments 1st Problem 1st Run										
Chandra Parallel Algorithm		Balachandran Linear Netork Model - AMPL								
Optimal path:		Total_Cost = 1501490				CPLEX 12.6.1.0: optimal solution; objective 1501492				
3 10 2 4 7 6 1 12 8 11 5 9 67930						132639 network simplex iterations.				
3 10 2 4 7 6 1 12 8 11 5 9 152930		Runtime in seconds:				0 simplex iterations (0 in phase I)				
3 10 2 4 7 6 1 12 8 11 5 9 237930		_ampl_user_time = 65.6562				Run time for solving the problem = 50.250000 seconds				
3 10 2 4 7 6 1 12 8 11 5 9 322930										
3 10 2 4 7 6 1 12 8 11 5 9 407930		Ship :=								
		0 1262 1		9 3 10 11 12 8 1 4 6 2 7 5						
Cost: 1273487		1262 3262 1		9 3 10 11 12 8 1 4 6 2 7 5						
		3262 5262 1		9 3 10 11 12 8 1 4 6 2 7 5						
real 1m34.856s		5262 7262 1		9 3 10 11 12 8 1 4 6 2 7 5						
user 7m6.958s		7262 9262 1		9 3 10 11 12 8 1 4 6 2 7 5						
sys 0m16.259s		9262 10001 1		9 3 10 11 12 8 1 4 6 2 7 5						

12 Departments 1st Problem 2nd Run										
Chandra Prallel Algorithm		Balachandran Linear Network Model - AMPL								
Optimal path:		Total_Cost = 1501180				CPLEX 12.6.1.0: optimal solution; objective 1501176				
8 1 12 11 10 3 5 9 2 6 7 4 43277						96542 network simplex iterations.				
8 1 12 11 10 3 5 9 2 6 7 4 128277		Runtime in seconds:				0 simplex iterations (0 in phase I)				
8 1 12 11 10 3 5 9 2 6 7 4 213277		_ampl_user_time = 68.2656				Run time for solving the problem = 53.125000 seconds				
8 1 12 11 10 3 5 9 2 6 7 4 298277										
8 1 12 11 10 3 5 9 2 6 7 4 383277		Ship :=								
		0 359 1		5 11 10 9 7 4 2 6 3 1 8 12						
Cost: 1252704		359 2359 1		5 11 10 9 7 4 2 6 3 1 8 12						
		2359 4359 1		5 11 10 9 7 4 2 6 3 1 8 12						
real 1m37.205s		4359 6359 1		5 11 10 9 7 4 2 6 3 1 8 12						
user 7m10.625s		6359 8359 1		5 11 10 9 7 4 2 6 3 1 8 12						
sys 0m15.760s		8359 10001 1		5 11 10 9 7 4 2 6 3 1 8 12						

12 Departments 1st Problem 3rd Run				
Chandra Parallel Algorithm		Balachandran Linear Network Model -AMPL		
Optimal path:		Total_Cost = 1418920		CPLEX 12.6.1.0: optimal solution; objective 1418924
11 8 12 1 6 2 4 7 5 3 10 9 77578				86114 network simplex iterations.
11 8 12 1 6 2 4 7 5 3 10 9 162578		Runtime in seconds:		0 simplex iterations (0 in phase I)
11 8 12 1 6 2 4 7 5 3 10 9 247578		_ampl_user_time = 65.8125		Run time for solving the problem = 50.640625 seconds
11 8 12 1 6 2 4 7 5 3 10 9 332578				
11 8 12 1 6 2 4 7 5 3 10 9 417578		Ship :=		
		0 1462 1	10 6 7 12 1 4 2 8 11 9 5 3	
Cost: 1278564		1462 3462 1	10 6 7 12 1 4 2 8 11 9 5 3	
		3462 5462 1	10 6 7 12 1 4 2 8 11 9 5 3	
real 1m35.907s		5462 7462 1	10 6 7 12 1 4 2 8 11 9 5 3	
user 7m6.441s		7462 9462 1	10 6 7 12 1 4 2 8 11 9 5 3	
sys 0m15.719s		9462 10001 1	10 6 7 12 1 4 2 8 11 9 5 3	

12 Departments 1st Problem 4th Run				
Chandra Parallel Algorithm		Balachandran Linear Network Model -AMPL		
Optimal path:		Total_Cost = 1455610		CPLEX 12.6.1.0: optimal solution; objective 1455606
11 3 10 8 12 1 4 7 2 6 9 5 13295				74458 network simplex iterations.
11 3 10 8 12 1 4 7 2 6 9 5 98295		Runtime in seconds:		0 simplex iterations (0 in phase I)
11 3 10 8 12 1 4 7 2 6 9 5 183295		_ampl_user_time = 59.8438		Run time for solving the problem = 45.984375 seconds
12 8 1 11 2 4 7 6 10 9 3 5 339132				
12 8 1 11 2 4 7 6 10 9 3 5 424132		Ship :=		
		0 542 1	5 6 9 3 7 12 11 4 8 1 10 2	
Cost: 1278994		542 2542 1	5 6 9 3 7 12 11 4 8 1 10 2	
		2542 4542 1	5 6 9 3 7 12 11 4 8 1 10 2	
real 1m39.069s		4542 6542 1	5 6 9 3 7 12 11 4 8 1 10 2	
user 7m9.074s		6542 8542 1	5 6 9 3 7 12 11 4 8 1 10 2	
sys 0m15.647s		8542 10001 1	5 6 9 3 7 12 11 4 8 1 10 2	

12th Departments 1st Problem 5th Run				
Chandra Parallel Algorithm		Balachandran Linear Network Model -AMPL		
Optimal path:		Total_Cost = 1350250		CPLEX 12.6.1.0: optimal solution; objective 1350247
1 12 4 7 2 6 11 9 10 5 3 8 56219				71258 network simplex iterations.
6 7 4 2 1 8 12 11 10 3 9 5 95807		Runtime in seconds:		0 simplex iterations (0 in phase I)
6 7 4 2 1 8 12 11 10 3 9 5 180807		_ampl_user_time = 63.2656		Run time for solving the problem = 48.406250 seconds
6 7 4 2 1 8 12 11 10 3 9 5 265807				
6 7 4 2 1 8 12 11 10 3 9 5 350807		Ship :=		
		0 1833 1	11 5 9 10 3 8 1 12 6 2 4 7	
Cost: 1259654		1833 3833 1	11 5 9 10 3 8 1 12 6 2 4 7	
		3833 5833 1	11 5 9 10 3 8 1 12 6 2 4 7	
real 1m35.490s		5833 7833 1	11 5 9 10 3 8 1 12 6 2 4 7	
user 7m6.870s		7833 9833 1	11 5 9 10 3 8 1 12 6 2 4 7	
sys 0m15.799s		9833 10001 1	11 5 9 10 3 8 1 12 6 2 4 7	

15 Departments 1st Problem 1st Run				
Chandra Parallel Algorithm		Balachandran Linear Network Model - AMPL		
Optimal path:		Total_Cost = 509933		CPLEX 12.6.1.0: optimal solution; objective 509933
7 14 13 10 2 5 8 15 11 3 4 1 12 9 6 73341				97397 network simplex iterations.
12 15 6 14 11 2 8 10 9 4 3 1 7 5 13 152011		Runtime in seconds:		0 simplex iterations (0 in phase I)
12 15 6 14 11 2 8 10 9 4 3 1 7 5 13 237011		_ampl_user_time = 61.2344		Run time for solving the problem = 46.640625 seconds
12 15 6 14 11 2 8 10 9 4 3 1 7 5 13 322011				
11 15 12 3 9 8 1 2 14 4 6 10 7 5 13 376680		Ship :=		
		0 906 1	4 9 12 2 3 5 15 13 11 1 7 8 14 6 10	
Cost: 502383		906 3217 1	4 9 12 2 3 5 15 13 11 1 7 8 14 6 10	
		3217 5217 1	4 8 9 14 3 15 6 10 7 5 11 2 12 1 13	
real 6m35.699s		5217 7217 1	4 8 9 14 3 15 6 10 7 5 11 2 12 1 13	
user 37m53.374s		7217 9583 1	11 3 10 8 9 4 2 5 7 13 15 14 12 1 6	
sys 0m39.218s		9583 10001 1	11 3 10 8 9 4 2 5 7 13 15 14 12 1 6	

15 Departments 1st Problem 2nd Run			
Chandra Parallel Algorithm	Balachandran Linear Network Model - AMPL		
Optimal path:	Total_Cost = 511825		CPLEX 12.6.1.0: optimal solution; objective 511825
10 6 4 15 7 5 12 13 11 3 1 2 8 14 9 29109			100476 network simplex iterations.
3 12 6 1 4 11 9 10 8 15 5 2 7 14 13 147348	Runtime in seconds:		0 simplex iterations (0 in phase I)
3 12 6 1 4 11 9 10 8 15 5 2 7 14 13 232348	_ampl_user_time = 64.1562		Run time for solving the problem = 50.296875 seconds
3 12 6 1 4 11 9 10 8 15 5 2 7 14 13 317348			
6 15 10 1 8 11 5 2 12 13 7 14 3 4 9 405874	Ship :=		
	0 95 1	3 5 13 10 6 4 14 12 9 8 2 15 7 11 1	
Cost: 503061	95 3047 1	3 1 9 15 13 6 14 4 12 10 11 5 8 2 7	
	3047 5047 1	3 1 9 15 13 6 14 4 12 10 11 5 8 2 7	
real 6m35.653s	5047 7047 1	3 1 9 15 13 6 14 4 12 10 11 5 8 2 7	
user 37m43.889s	7047 9719 1	3 1 9 15 13 6 14 4 12 10 11 5 8 2 7	
sys 0m37.575s	9719 10001 1	6 9 14 12 15 8 5 2 13 11 4 1 7 10 3	

15 Departments 1st Problem 3rd Run			
Chandra Parallel Algorithm	Balachandran Linear Network Model -AMPL		
Optimal path:	Total_Cost = 512736		CPLEX 12.6.1.0: optimal solution; objective 512736
7 9 6 12 2 5 11 15 13 4 1 8 3 14 10 24467			111902 network simplex iterations.
13 3 1 15 2 12 10 8 6 4 11 9 7 5 14 95963	Runtime in seconds:		0 dual simplex iterations (0 in phase I)
13 3 1 15 2 12 10 8 6 4 11 9 7 5 14 180963	_ampl_user_time = 75.5825		Run time for solving the problem = 54.257148 seconds
13 3 1 15 2 12 10 8 6 4 11 9 7 5 14 265963			
10 1 6 15 8 12 4 2 5 7 11 9 3 14 13 410389	Ship :=		
	0 801 1	7 15 8 6 3 4 11 13 12 9 5 10 14 1 2	
Cost: 501022	801 2376 1	7 15 8 6 3 4 11 13 12 9 5 10 14 1 2	
	2376 4376 1	4 14 8 2 13 12 7 6 9 1 11 5 3 10 15	
real 6m37.249s	4376 6376 1	4 14 8 2 13 12 7 6 9 1 11 5 3 10 15	
user 37m53.206s	6376 8858 1	4 14 8 2 13 12 7 6 9 1 11 5 3 10 15	
sys 0m37.469s	8858 10001 1	8 11 5 4 9 7 1 6 2 10 12 13 3 14 15	

15 Departments 1st Problem 4th Run			
Chandra Parallel Algorithm	Balachandran Linear Network Model - AMPL		
Optimal path:	Total_Cost = 511302		CPLEX 12.6.1.0: optimal solution; objective 511302
4 8 15 3 2 7 11 13 14 5 6 9 12 10 1 11525			101290 network simplex iterations.
4 5 6 10 15 13 9 8 7 1 11 12 2 14 3 107614	Runtime in seconds:		0 simplex iterations (0 in phase I)
4 5 6 10 15 13 9 8 7 1 11 12 2 14 3 192614	_ampl_user_time = 61.375		Run time for solving the problem = 47.046875 seconds
4 5 6 10 15 13 9 8 7 1 11 12 2 14 3 277614			
10 14 6 4 9 15 5 2 3 13 7 12 1 11 8 419001	Ship :=		
	0 1376 1	10 4 6 2 1 12 13 15 9 3 5 14 11 8 7	
Cost: 501945	1376 3821 1	10 4 6 2 1 12 13 15 9 3 5 14 11 8 7	
	3821 5821 1	3 5 7 2 11 12 6 10 9 1 4 14 13 8 15	
real 6m38.987s	5821 7821 1	3 5 7 2 11 12 6 10 9 1 4 14 13 8 15	
user 38m10.021s	7821 9141 1	3 5 7 2 11 12 6 10 9 1 4 14 13 8 15	
sys 0m37.777s	9141 10001 1	7 5 15 8 11 14 2 3 13 4 6 10 12 1 9	

15 Departments 1st Problem 5th Run			
Chandra Parallel Algorithm	Balachandran Linear Network Model -AMPL		
Optimal path:	Total_Cost = 509640		
9 11 6 3 10 8 12 15 13 7 4 5 14 1 2 2458			CPLEX 12.6.1.0: optimal solution; objective 509640
14 4 6 3 12 5 9 10 7 11 13 1 8 15 2 150014	Runtime in seconds:		90641 network simplex iterations.
14 4 6 3 12 5 9 10 7 11 13 1 8 15 2 235014	_ampl_user_time = 60.9219		0 simplex iterations (0 in phase I)
14 4 6 3 12 5 9 10 7 11 13 1 8 15 2 320014			Run time for solving the problem = 46.015625 seconds
9 4 5 12 8 13 3 2 1 6 11 10 14 15 7 414740	Ship :=		
	0 1583 1	1 5 13 2 4 6 11 12 15 8 7 9 14 10 3	
Cost: 499324	1583 3742 1	1 5 13 2 4 6 11 12 15 8 7 9 14 10 3	
	3742 5742 1	11 1 15 2 12 4 7 8 6 3 5 10 9 14 13	
real 6m34.729s	5742 7742 1	11 1 15 2 12 4 7 8 6 3 5 10 9 14 13	
user 37m42.764s	7742 9543 1	11 1 15 2 12 4 7 8 6 3 5 10 9 14 13	
sys 0m37.746s	9543 10001 1	7 5 14 6 11 9 3 2 12 4 13 10 1 15 8	

30 Departments 1st Problem 1st Run								
Chandra Parallel Algorithm								
Optimal path:								
1 5 18 24 14 27 11 6 23 29 26 10 12 16 22 3 28 17 30 15 13 25 21 7 20 8 9 19 2 4 58000								
1 5 18 24 14 27 11 6 23 29 26 10 12 16 22 3 28 17 30 15 13 25 21 7 20 8 9 19 2 4 143000								
1 5 18 24 14 27 11 6 23 29 26 10 12 16 22 3 28 17 30 15 13 25 21 7 20 8 9 19 2 4 228000								
1 5 18 24 14 27 11 6 23 29 26 10 12 16 22 3 28 17 30 15 13 25 21 7 20 8 9 19 2 4 313000								
1 5 18 24 14 27 11 6 23 29 26 10 12 16 22 3 28 17 30 15 13 25 21 7 20 8 9 19 2 4 398000								
Cost: 601677								
real 7m35.407s								
user 42m12.088s								
sys 0m40.262s								
Balachandran Linear Network Model -AMPL								
Total_Cost = 610863		CPLEX 12.6.1.0: optimal solution; objective 610863						
		65260 network simplex iterations.						
Runtime in seconds:		0 simplex iterations (0 in phase I)						
_ampl_user_time = 62.1719		Run time for solving the problem = 47.437500 seconds						
Ship :=								
0 1521 1	1 7 30 13 28 9 19 16 29 11 18 8 4 21 24 17 15 2 14 20 25 22 12 10 26 27 6 23 5 3							
1521 3521 1	1 7 30 13 28 9 19 16 29 11 18 8 4 21 24 17 15 2 14 20 25 22 12 10 26 27 6 23 5 3							
3521 5521 1	1 7 30 13 28 9 19 16 29 11 18 8 4 21 24 17 15 2 14 20 25 22 12 10 26 27 6 23 5 3							
5521 7521 1	1 7 30 13 28 9 19 16 29 11 18 8 4 21 24 17 15 2 14 20 25 22 12 10 26 27 6 23 5 3							
7521 9521 1	1 7 30 13 28 9 19 16 29 11 18 8 4 21 24 17 15 2 14 20 25 22 12 10 26 27 6 23 5 3							
9521 10001 1	1 7 30 13 28 9 19 16 29 11 18 8 4 21 24 17 15 2 14 20 25 22 12 10 26 27 6 23 5 3							

30 Departments 1st Problem 2nd Run								
Chandra Parallel Algorithm								
Optimal path:								
7 16 22 2 24 18 1 28 15 30 20 5 10 13 29 17 25 12 9 21 23 14 6 8 4 3 11 26 19 27 18893								
7 16 22 2 24 18 1 28 15 30 20 5 10 13 29 17 25 12 9 21 23 14 6 8 4 3 11 26 19 27 103893								
7 16 22 2 24 18 1 28 15 30 20 5 10 13 29 17 25 12 9 21 23 14 6 8 4 3 11 26 19 27 188893								
7 16 22 2 24 18 1 28 15 30 20 5 10 13 29 17 25 12 9 21 23 14 6 8 4 3 11 26 19 27 273893								
7 16 22 2 24 18 1 28 15 30 20 5 10 13 29 17 25 12 9 21 23 14 6 8 4 3 11 26 19 27 358893								
Cost: 602742								
real 7m37.846s								
user 42m41.713s								
sys 0m39.510s								
Balachandran Linear Network Model -AMPL								
Total_Cost = 609194								
				CPLEX 12.6.1.0: optimal solution; objective 609194				
Runtime in seconds:				59679 network simplex iterations.				
_ampl_user_time = 60.2656				0 simplex iterations (0 in phase I)				
				Run time for solving the problem = 45.984375 seconds				
Ship :=								
0 269 1	20 10 13 29 11 3 17 25 21 22 12 27 28 19 1 15 23 14 16 26 2 30 6 18 5 24 4 8 7 9							
269 2269 1	20 10 13 29 11 3 17 25 21 22 12 27 28 19 1 15 23 14 16 26 2 30 6 18 5 24 4 8 7 9							
2269 4269 1	20 10 13 29 11 3 17 25 21 22 12 27 28 19 1 15 23 14 16 26 2 30 6 18 5 24 4 8 7 9							
4269 6269 1	20 10 13 29 11 3 17 25 21 22 12 27 28 19 1 15 23 14 16 26 2 30 6 18 5 24 4 8 7 9							
6269 8269 1	20 10 13 29 11 3 17 25 21 22 12 27 28 19 1 15 23 14 16 26 2 30 6 18 5 24 4 8 7 9							
8269 10001 1	20 10 13 29 11 3 17 25 21 22 12 27 28 19 1 15 23 14 16 26 2 30 6 18 5 24 4 8 7 9							

30 Departments 1st Problem 3rd Run	
Chandra Parallel Algorithm	
Optimal path:	
6 27 9 16 26 10 20 19 28 24 25 7 17 15 29 21 23 8 2 30 5 22 1 13 12 4 14 18 3 11 79260	
6 27 9 16 26 10 20 19 28 24 25 7 17 15 29 21 23 8 2 30 5 22 1 13 12 4 14 18 3 11 164260	
6 27 9 16 26 10 20 19 28 24 25 7 17 15 29 21 23 8 2 30 5 22 1 13 12 4 14 18 3 11 249260	
6 27 9 16 26 10 20 19 28 24 25 7 17 15 29 21 23 8 2 30 5 22 1 13 12 4 14 18 3 11 334260	
6 27 9 16 26 10 20 19 28 24 25 7 17 15 29 21 23 8 2 30 5 22 1 13 12 4 14 18 3 11 419260	
Cost: 603472	
real 7m36.142s	
user 42m12.811s	
sys 0m39.824s	
Balachandran Linear Network Model - AMPL	
Total_Cost = 608418	
	CPLEX 12.6.1.0: optimal solution; objective 608418
Runtime in seconds:	82167 network simplex iterations.
_ampl_user_time = 60.4844	0 simplex iterations (0 in phase I)
	Run time for solving the problem = 46.500000 seconds
Ship :=	
0 550 1	10 25 18 19 30 9 20 23 15 17 6 3 26 21 13 28 2 22 12 14 29 5 24 1 7 11 4 27 16 8
550 2550 1	10 25 18 19 30 9 20 23 15 17 6 3 26 21 13 28 2 22 12 14 29 5 24 1 7 11 4 27 16 8
2550 4550 1	10 25 18 19 30 9 20 23 15 17 6 3 26 21 13 28 2 22 12 14 29 5 24 1 7 11 4 27 16 8
4550 6550 1	10 25 18 19 30 9 20 23 15 17 6 3 26 21 13 28 2 22 12 14 29 5 24 1 7 11 4 27 16 8
6550 8550 1	10 25 18 19 30 9 20 23 15 17 6 3 26 21 13 28 2 22 12 14 29 5 24 1 7 11 4 27 16 8
8550 10001 1	10 25 18 19 30 9 20 23 15 17 6 3 26 21 13 28 2 22 12 14 29 5 24 1 7 11 4 27 16 8

30 Departments 1st Problem 4th Run	
Chandra Parallel Algorithm	
Optimal path:	
2 12 1 19 8 20 16 18 29 26 5 10 4 3 13 24 22 27 11 30 25 15 23 17 7 6 21 28 14 9 14152	
2 12 1 19 8 20 16 18 29 26 5 10 4 3 13 24 22 27 11 30 25 15 23 17 7 6 21 28 14 9 99152	
2 12 1 19 8 20 16 18 29 26 5 10 4 3 13 24 22 27 11 30 25 15 23 17 7 6 21 28 14 9 184152	
2 12 1 19 8 20 16 18 29 26 5 10 4 3 13 24 22 27 11 30 25 15 23 17 7 6 21 28 14 9 269152	
2 12 1 19 8 20 16 18 29 26 5 10 4 3 13 24 22 27 11 30 25 15 23 17 7 6 21 28 14 9 354152	
Cost: 603722	
real 7m36.883s	
user 42m35.264s	
sys 0m39.933s	
Balachandran Linear Network Model -AMPL	
Total_Cost = 609015	
	CPLEX 12.6.1.0: optimal solution; objective 609015
Runtime in seconds:	60940 network simplex iterations.
_ampl_user_time = 59.5312	0 simplex iterations (0 in phase I)
	Run time for solving the problem = 45.640625 seconds
Ship :=	
0 434 1	11 3 9 21 4 8 19 15 23 25 22 7 13 29 28 6 14 20 2 5 17 1 24 18 10 26 12 30 27 16
434 2434 1	11 3 9 21 4 8 19 15 23 25 22 7 13 29 28 6 14 20 2 5 17 1 24 18 10 26 12 30 27 16
2434 4434 1	11 3 9 21 4 8 19 15 23 25 22 7 13 29 28 6 14 20 2 5 17 1 24 18 10 26 12 30 27 16
4434 6434 1	11 3 9 21 4 8 19 15 23 25 22 7 13 29 28 6 14 20 2 5 17 1 24 18 10 26 12 30 27 16
6434 8434 1	11 3 9 21 4 8 19 15 23 25 22 7 13 29 28 6 14 20 2 5 17 1 24 18 10 26 12 30 27 16
8434 10001 1	11 3 9 21 4 8 19 15 23 25 22 7 13 29 28 6 14 20 2 5 17 1 24 18 10 26 12 30 27 16

30 Departments 1st Problem 5th Run									
Chandra Parallel Algorithm									
Optimal path:									
27 24 29 28 18 10 7 1 23 15 30 2 5 21 12 26 17 11 16 19 22 25 13 14 6 3 8 20 9 4 74873									
27 24 29 28 18 10 7 1 23 15 30 2 5 21 12 26 17 11 16 19 22 25 13 14 6 3 8 20 9 4 159873									
27 24 29 28 18 10 7 1 23 15 30 2 5 21 12 26 17 11 16 19 22 25 13 14 6 3 8 20 9 4 244873									
27 24 29 28 18 10 7 1 23 15 30 2 5 21 12 26 17 11 16 19 22 25 13 14 6 3 8 20 9 4 329873									
27 24 29 28 18 10 7 1 23 15 30 2 5 21 12 26 17 11 16 19 22 25 13 14 6 3 8 20 9 4 414873									
Cost: 604214									
real 7m38.750s									
user 42m43.590s									
sys 0m39.974s									
Balachandran Linear Network Model - AMPL									
Total_Cost = 609364									
				CPLEX 12.6.1.0: optimal solution; objective 609364					
Runtime in seconds:				63519 network simplex iterations.					
_ampl_user_time = 79.2641				0 dual simplex iterations (0 in phase I)					
				Run time for solving the problem = 58.016772 seconds					
Ship :=									
0 1707 1	6 3 23 20 9 8 24 21 25 15 18 7 11 16 19 13 28 4 27 1 22 12 30 2 17 14 29 5 26 10								
1707 3707 1	6 3 23 20 9 8 24 21 25 15 18 7 11 16 19 13 28 4 27 1 22 12 30 2 17 14 29 5 26 10								
3707 5707 1	6 3 23 20 9 8 24 21 25 15 18 7 11 16 19 13 28 4 27 1 22 12 30 2 17 14 29 5 26 10								
5707 7707 1	6 3 23 20 9 8 24 21 25 15 18 7 11 16 19 13 28 4 27 1 22 12 30 2 17 14 29 5 26 10								
7707 9707 1	6 3 23 20 9 8 24 21 25 15 18 7 11 16 19 13 28 4 27 1 22 12 30 2 17 14 29 5 26 10								
9707 10001 1	6 3 23 20 9 8 24 21 25 15 18 7 11 16 19 13 28 4 27 1 22 12 30 2 17 14 29 5 26 10								

APPENDIX J: Simplex iterations *LNM* solved in AMPL - 6 departments

Problem Number of departments_problem number_ run number	*Sorting*	*No-sorting*
6_1_1	24077	24972
6_1_2	24077	24972
6_1_3	24077	24972
6_1_4	24077	24972
6_1_5	24077	24972
6_2_1	19233	24277
6_2_2	19233	24277
6_2_3	19233	24277
6_2_4	19233	24277
6_2_5	19233	24277
6_3_1	21430	23808
6_3_2	21430	23808
6_3_3	21430	23808
6_3_4	21430	23808
6_3_5	21430	23808
6_4_1	22528	26053
6_4_2	22528	26053
6_4_3	22528	26053
6_4_4	22528	26053
6_4_5	22528	26053
6_5_1	21568	22278
6_5_2	21568	22278
6_5_3	21568	22278
6_5_4	21568	22278
6_5_5	21568	22278
6_6_1	20413	24187
6_6_2	20413	24187
6_6_3	20413	24187
6_6_4	20413	24187
6_6_5	20413	24187
6_7_1	20796	20097
6_7_2	20796	20097
6_7_3	20796	20097
6_7_4	20796	20097
6_7_5	20796	20097
6_8_1	25886	28705
6_8_2	25886	28705
6_8_3	25886	28705
6_8_4	25886	28705
6_8_5	25886	28705

APPENDIX K: Simplex iterations *LNM* solved in AMPL - 15 departments

Problem Number of departments_problem number_ run number	*Sorting*	*No- sorting*
15_1_1	81002	97397
15_1_2	78272	100476
15_1_3	63112	111902
15_1_4	83060	101290
15_1_5	64469	90641
15_2_1	54170	106389
15_2_2	65007	71097
15_2_3	86644	58733
15_2_4	65043	119468
15_2_5	83704	99292
15_3_1	61579	80660
15_3_2	62807	83790
15_3_3	57092	76346
15_3_4	83704	105136
15_3_5	53195	102944
15_4_1	57716	92498
15_4_2	49208	107528
15_4_3	51300	96242
15_4_4	67496	82229
15_4_5	65228	74709
15_5_1	59238	107148
15_5_2	94007	91955
15_5_3	50953	85983
15_5_4	48668	90162
15_5_5	48497	98718
15_6_1	59404	130186
15_6_2	56373	66529
15_6_3	53270	103436
15_6_4	45773	90543
15_6_5	65619	96583
15_7_1	63025	114633
15_7_2	47440	94429
15_7_3	53514	81625
15_7_4	64860	84925
15_7_5	53815	95910
15_8_1	77389	93352
15_8_2	51431	92172
15_8_3	61620	86898
15_8_4	59069	111351
15_8_5	53236	80978

APPENDIX L: Simplex iterations *LNM* solved in AMPL - 30 departments

Problem Number of departments_problem number_ run number	*Sorting*	*No-sorting*
30_1_1	91052	65260
30_1_2	70818	59679
30_1_3	56930	82167
30_1_4	72040	60940
30_1_5	68203	63519
30_2_1	72421	81854
30_2_2	141896	75925
30_2_3	72850	88339
30_2_4	61028	65568
30_2_5	87913	76070
30_3_1	87913	67162
30_3_2	61980	74932
30_3_3	61980	56268
30_3_4	62711	80280
30_3_5	93690	69650
30_4_1	74625	75723
30_4_2	63769	68761
30_4_3	50258	71067
30_4_4	50638	80099
30_4_5	72403	85473
30_5_1	49323	80500
30_5_2	88280	118737
30_5_3	55895	56139
30_5_4	55807	248608
30_5_5	46001	114839
30_6_1	58509	113537
30_6_2	67823	99685
30_6_3	84082	95670
30_6_4	323556	64152
30_6_5	44423	128863
30_7_1	133985	66140
30_7_2	96465	89949
30_7_3	78469	72823
30_7_4	277163	77819
30_7_5	59042	78040
30_8_1	94777	112548
30_8_2	77368	89997
30_8_3	100458	57206
30_8_4	71470	59438
30_8_5	66207	49142

APPENDIX M: Table 58: Relevant Information about the Computational Environments Used

Problem or Algorithm	Cluster or computer	Operating System	Speed (Ghz)	CPU's	Memory (GB)
PSP, LNM no-*sorting*	Stampede cluster	CentOS	2.8	20	12.8
PSP, LNM sorting	Maverick cluster	CentOS	2.7	10	32
LNM for DFLP *sorting* and no-*sorting*, *SDFLP*	SOLAR lab PC	Windows	3.2	4	16
SDFLP	LEAP	CentOS	2.4	28	128
LNM sorting, *PSP sorting*	CAPI	CentOS	2.06 - 3.69	160	16

APPENDIX N: Table 59: Costs, Computational Times and Percentage Difference (% Diff) for *PSP* and *LNM – Sorting* (S) vs. *No-Sorting* (NS) - 15 Departments - Times Shortest Path Sorting (SP-S) are only for doing the variant of the Dijkstra's Algorithm

PSP-S	*PSP-NS*	% Diff	*LNM -S*	*LNM-NS*	% Diff
498273	501547	0.65	501547	511087	1.9
499297	502998	0.74	502998	512633	1.9
500066	506883	**1.34**	507031	515212	1.59
498717	501035	0.46	501035	508173	1.4
499414	500312	0.18	500312	511144	**2.12**
499353	502486	0.62	502486	511498	1.76
501169	504100	0.58	504100	514077	1.94
505761	507402	0.32	507442	517778	2
SP-S	*PSP-NS*	% Diff	*LNM-S*	*LNM-NS*	% Diff
41.2	36.5	-12.88	45.7	64.7	29
41.6	97.4	57.29	45.7	61.1	25
42.4	396.8	89.31	47.8	62.1	22.99
42.4	396.1	89.3	46	60.3	25.42
42	396.4	89.4	45.2	60.4	25.1
46.6	396.1	88.24	45.3	60.9	25.56
41.2	395.8	89.59	45.4	60.3	24.74
41.4	384.1	89.22	45.3	59.9	24.34

Table 60: Costs, Computational Times and Percentage Difference (% Diff) for *PSP* and *LNM – Sorting* (S) vs. *No-Sorting* (NS) - 30 Departments - Times Shortest Path Sorting (SP-S) are only for doing the variant of the Dijkstra's Algorithm

PSP-S	*PSP-NS*	% Diff	*LNM -S*	*LNM-NS*	% Diff
603233	603165	-0.01	624464	609371	-2.48
603332	600408	-0.49	606909	606746	-0.03
605783	607720	0.32	612155	611773	-0.06
597999	601261	0.54	601721	607288	0.92
587107	590454	0.57	590548	599579	1.51
594994	597884	0.48	598842	603318	0.74
597179	595900	-0.21	599936	601822	0.31
601663	603972	0.38	603848	614843	1.79

Table 60 (Continued)

SP-S	*PSP-NS*	% Diff	*LNM-S*	*LNM-NS*	% Diff
41.6	457	90.9	47.7	64.3	25.86
41.2	457.4	90.99	47.9	71.9	33.32
41.8	456.4	90.84	47.9	60.8	21.21
41.6	106.2	60.81	48.5	60	19.23
41.2	107.8	61.76	48.8	60.1	18.81
41	107.9	61.99	48.4	77.6	37.71
41	111.2	63.13	45.7	60.1	23.97
41.6	107.1	61.17	45.8	76.8	40.45

APPENDIX O: Table 61: Times to Perform only the variant of the Dijkstra algorithm for *SP Sorting* (S)- 15 and 30 departments

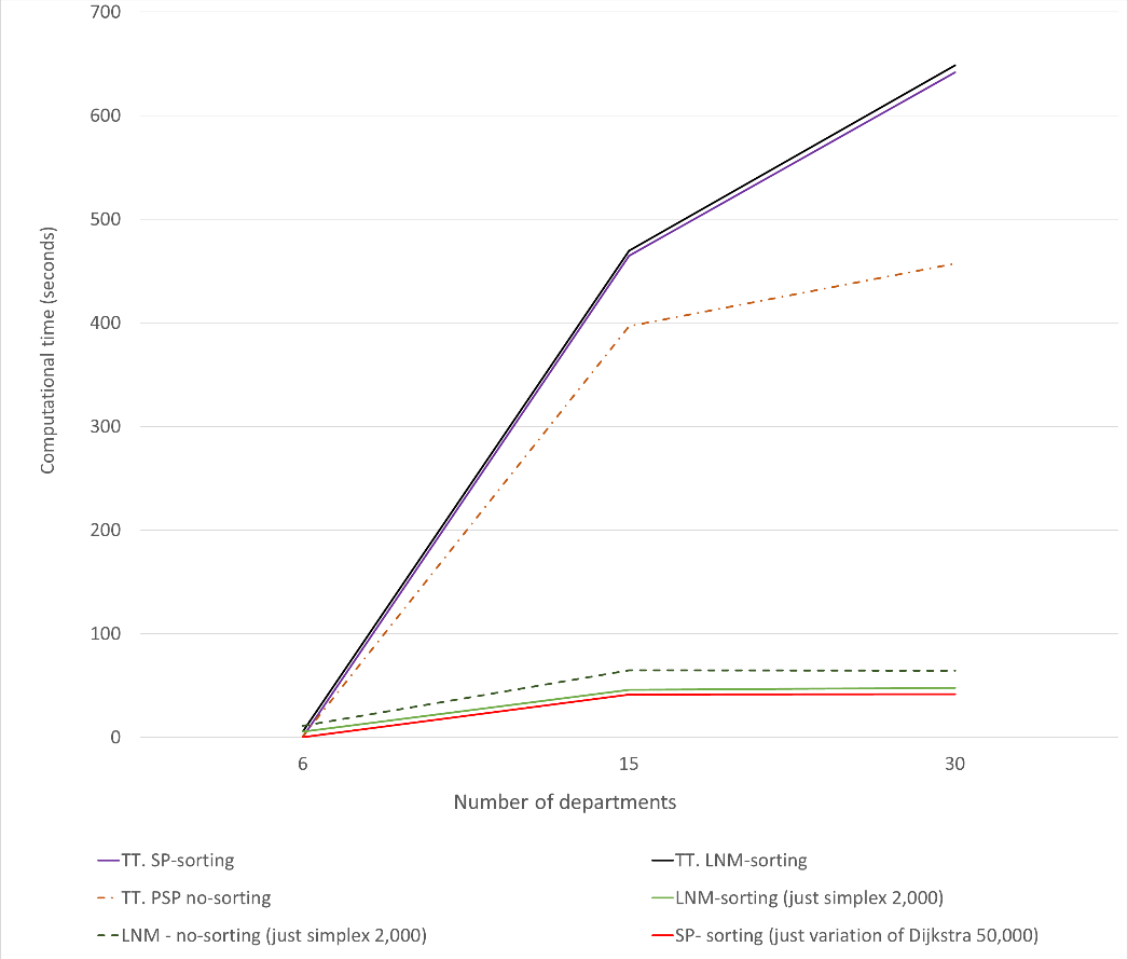
	Problem #	R1	R2	R3	R4	R5	Average	Std. Dev
15	1	41	41	42	41	41	41.2	0
	2	41	41	42	43	41	41.6	1
	3	42	42	42	43	43	42.4	1
	4	41	42	43	42	44	42.4	1
	5	42	41	42	41	44	42.0	1
	6	41	69	41	41	41	46.6	13
	7	41	41	41	42	41	41.2	0
	8	42	41	42	41	41	41.4	1
30	1	41	44	41	41	41	41.6	1
	2	41	41	41	41	42	41.2	0
	3	41	41	41	44	42	41.8	1
	4	41	41	41	41	44	41.6	1
	5	41	41	41	42	41	41.2	0
	6	41	41	41	41	41	41.0	0
	7	41	41	41	41	41	41.0	0
	8	43	42	41	41	41	41.6	1

APPENDIX P: Table 62: Times to Perform *LNM Sorting* (S) – 15 and 30 Departments including Percentage Difference (% Diff) for *LNM Sorting* (S) vs *SP Sorting* (S) as reported in Appendix O

	Problem	R1	R2	R3	R4	R5	Average	Std. Dev	% Diff *LNM* vs *SP*
15	1	47.31	45.45	45.07	45.10	45.71	45.728	0.923	9.90%
	2	45.00	45.48	45.29	45.73	46.86	45.672	0.716	8.92%
	3	45.85	46.17	46.26	46.81	53.96	47.810	3.455	11.32%
	4	45.43	45.28	44.62	44.89	44.70	44.984	0.357	5.74%
	5	45.00	44.93	44.64	46.20	45.40	45.234	0.604	7.15%
	6	45.96	45.48	45.10	44.71	45.37	45.324	0.463	-2.82%
	7	45.87	45.32	45.46	45.40	44.81	45.372	0.379	9.20%
	8	45.17	44.67	46.48	45.39	44.95	45.332	0.695	8.67%
30	1	45.75	47.31	49.28	48.92	47.26	47.704	1.426	12.80%
	2	47.81	47.54	48.96	47.53	47.87	47.942	0.590	14.06%
	3	47.87	48.40	48.21	47.48	47.42	47.876	0.433	12.69%
	4	48.46	48.50	48.71	48.48	48.25	48.480	0.163	14.19%
	5	48.42	48.14	49.39	49.57	48.29	48.762	0.666	15.51%
	6	48.39	48.23	47.75	45.54	49.01	47.784	1.333	14.20%
	7	46.21	45.46	45.53	45.89	45.45	45.708	0.333	10.30%
	8	45.64	45.20	46.10	45.65	46.26	45.770	0.420	9.11%

% Diff = 100*(*LNM sorting –SP sorting*)/*LNM sorting*

APPENDIX Q: Computational Time Graph for Various Methods Studied



REFERENCES

Azadivar, F., & Wang, J. (2000). Facility layout optimization using simulation and genetic algorithms. *International Journal of Production Research*, 38(17), 4369-4383.

Azimi. P, Charmchi, H. R. (2012). A new optimization approach for dynamic facility layout with budget constraint. *Operations Research*, 168 (2), 57–89.

Balakrishnan, J., Jacobs, R. F., & Venkataraman, M.A. (1992). Solutions for the constrained Dynamic Facility Layout Problem. *European Journal of Operations Research*, 57(2), 280-286.

Balakrishnan, J., & Cheng, C. H. (1998). Dynamic Layout Algorithms: A State-of-the-art Survey. *Omega International Journal of Management Science*, 26(4), 507-521.

Balakrishnan, J., & Cheng, C. H. (2000). Genetic search and the dynamic layout problem. *Computers & Operations Research*, 27(6), 587-593.

Balakrishnan, J., & Cheng, C. H., Conway, D.G., Lau, M.C. (2003). A hybrid genetic algorithm for the dynamic plant layout problem. *International Journal of Production Economics*, 86(2), 107–120.

Balakrishnan, J., & Cheng, C. H. (2006). A note on "a hybrid genetic algorithm for the dynamic plant layout problem." *International Journal of Production Economics*, 103(1), 87–89.

Balakrishnan, J., & Cheng, C. H. (2009). The dynamic plant layout problem: Incorporating rolling horizons and forecast uncertainty. *The international journal of management science, Omega*, 37(1) 165–177.

Balas, E., & Mazzola, J.B. (1980). Quadratic 0-1 programming by a new linearization, *TIMS/ORSA meeting*, Washington, DC.

Ballou, R. J. (2003). *Business Logistics/Supply Chain Management and Logware*. Pearson/Prentice Hall, Upper Sadle River, NJ.

Ballou, R. H. (1968). Dynamic Warehouse location analysis. *Journal of Marketing Research*, 5, 271-276.

Baykasoglu, A., Dereli, T., & Sabuncu, I. (2006). An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems, *Omega*, 34(4), 385–396.

Baykasoglu, A., & Gindy, N. N. Z. (2001). A simulated annealing algorithm for dynamic layout problem. *Computers & Operations Research*, 28(14), 1403-1426.

Benjaafar, S., & Sheikhzadeh, M. (1997). Design of flexible plant layouts. *IIE Transactions*, 32(4), 309-322.

Birge, J. R., Louveaux, F. (2010). *Introduction to Stochastic Programming.* Second Edition. Springer Series in Operations Research and Financial Engineering. Mikosch, T.V., Resnick, S.I., and Robinson, S.M., Editors. Springer, New York, NY.

Burkard, R. E., & Bonniger, T. (1983). A heuristic for quadratic Boolean program with applications to quadratic assignment problems. *European Journal of Operational Research*, 13(4), 374-386.

Bazaraa, S. M., Jarvis, J. J., & Sherali, H. D. (1990). *Linear programming and network flows- Second Edition*. John Wiley & Sons, *New* York, NY.

Conway, D.G., & Venkataramanan, M.A. (1994). Genetic search and the dynamic facility layout problem. *Computers & Operations Research*, 21(8), 955-960.

Dijkstra, E.W. (1959). A note on two problems in connection with graphs. *Number Math*. 1, 269-271.

Fourer, R, Gay, D.M., & Kernighan, B.W. (2003). *A Modeling Language for Mathematical Programming*, Thomson, Canada, Second edition, 319-351.

Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci Heaps and their uses in improved network optimization algorithms. *Journal of the association for computing machinery*, 34(3), 596-615.

Kaku, B.K., & Mazzola, J. B. (1997). A Tabu search heuristic for the dynamic plant layout problem. *INFORMS Journal on Computing*, 9(4), 374-384.

Kolla, C.S. (2015). Finding Efficient Solutions to the Dynamic Plant Layout Problem with Dijkstra's Shortest Path Algorithm. Independent Study. Department of Computer Science. Texas State University.

Krishnan, K. K., Cheraghi, H. S., & Nayak, C.N. (2008). Facility layout design for multiple production scenarios in a dynamic environment. *International Journal of Industrial and Systems Engineering*, 3(2), 105-133.

Kulturel-Konak, S. (2007). Approaches to uncertainties in facility layout problems: perspectives at the beginning of the 21st century. *Journals of Intelligent Manufacturing*, 18(2), 273–284.

Lacksonen, T. A., & Enscore, E. E. (1993). Quadratic assignment algorithms for the dynamic layout problem. *International Journal of Production Research*, 31(3), 503-517.

Leap: High Performance Computing Cluster. Texas State University. Retrieved from http://www.vpit.txstate.edu/rc/leap.html

Loiola, E. M., De Abreu, N., Boaventura Netto, P., Hahn, P. & Querido, T. (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2), 657–690.

McKendall, R. A., & Shang, J. (2006). Hybrid ant systems for the dynamic facility layout problem. *Computers & Operations Research*, 33(3), 790 – 803.

MicroPower. Retrieved from http://MicroPower-global.com

Moselimipour, G., Lee, T.S., & Rilling, D. (2012). A review of intelligent approaches for designing dynamic and robust layouts in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 60(1-4), 11-27.

Moslemipour, G., & Lee, T.S. (2012). Intelligent design of a dynamic machine layout in uncertain environment of flexible manufacturing systems. *Journals of Intelligent Manufacturing*, 23(5), 1-12.

Novoa, C. M., & Mai, N. (2013). Facilities Layout at MicroPower. In *Decision Sciences Institute 44th Annual Meeting* (pp. 671963-1-14).

Palekar, S. U., Batta, R., Bosch, R.M., & Elhence, S. (1992). Modeling uncertainties in plant layout problems. *European Journal of Operational Research*, 63(2), 347-359.

Rosenblatt, M. J. (1986). The dynamics of plant layout. *Management Science*, 32(1),76-86.

Rardin, R. (2017). *Optimization in Operations Research Second Edition*. Pearson, Hoboken, NJ.

Sahin, R., Ertogral, K. & Turkbey, O. (2010). A simulated annealing heuristic for the dynamic layout problem with budget constraint. *Computers & Industrial Engineering*, 59, 308–313.

Sweeney, D. S. & Tatham, R. L. (1976). An improved long run model for multiple warehouse location. *Management Science*, 22(7), 758-758.

Taha, H. A. (2013). *Operations Research – An Introduction Ninth Edition*. Prentice Hall, Upper Saddle River, NJ.

Tarjan, R. E. (1983). *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Tayal, A., Gunasekaran, A., Singh, S. P., Dubey, R., & Papadopoulos, T. (2017). Formulating and solving sustainable dynamic facility layout problem: a key to sustainable operations. *Annals of Operations Research*. 253, 621-655. https://doi.org/10.1007/s10479-016-2351-9

Tayal, A., Singh, S. P. (2018). Formulating multi objective stochastic dynamic facility layout problem for disaster relief. *Annals of Operations Research.* https://doi.org/10.1007/s10479-017-2592-2

Texas Advanced Computing Center (TACC). University of Texas at Austin. Pickle Research Center. https://portal.tacc.utexas.edu

Urban, L. T. (1998). Solution procedures for the dynamic facility layout problem. *Journal of Operations Research*, 76(0), 323-342.

Vitayasak, S., Pongcharoen, P., & Hicks, C. (2017). A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm. *International Journal of Production Economics*, 190C, 146-157.

Winston, W. L. (2004*). Operations Research – Applications and Algorithms Fourth Edition*. Brooks/Cole, Cengage Learning, Belmont, CA.