THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Doctor of Engineering (Industrial Engineering)
DEGREE

Industrial Engineering
FIELD

Industrial Engineering
DEPARTMENT

TITLE: Benders' Decomposition Based Heuristics for the Dynamic Quadratic Assignment Problem

NAME: Ms. Sirirat Muenvanichakul

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR
(Associate Professor Peerayuth Charnsethikul, Ph.D.)

COMMITTEE MEMBER
(Associate Professor Vira Chankong, Ph.D.)

COMMITTEE MEMBER
(Associate Professor Prapaisri Sudasana-na-Ayudthya, Ph.D.)

COMMITTEE MEMBER
(Associate Professor Saeree Svetasreni, Ph.D.)

COMMITTEE MEMBER
(Associate Professor Santi Wiriyawit, Ph.D.)

DEPARTMENT HEAD
(Associate Professor Anan Mungwattana, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON _____

DEAN
(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

BENDERS' DECOMPOSITION BASED HEURISTICS FOR THE DYNAMIC QUADRATIC ASSIGNMENT PROBLEM

SIRIRAT MUENVANICHAKUL

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Engineering (Industrial Engineering)
Graduate School, Kasetsart University
2009

Sirirat Muenvanichakul 2009: Benders' Decomposition Based Heuristics for the Dynamic Quadratic Assignment Problem. Doctor of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering. Thesis Advisor: Associate Professor Peerayuth Charnsethikul, Ph.D. 128 pages.

The objective of the dynamic quadratic assignment problem (DQAP) is to minimize the sum of the assignment cost and the rearrangement cost over all periods. As a result, DQAP is an NP-hard problem and it is extremely difficult to determine an optimal solution of DQAP. This thesis has focused on developing optimization methods to approximate the exact solution of large-scale DQAPs. Three approaches are taken – Benders' decomposition (BD) based method, dynamic programming (DP) and logic-based method.

Benders' decomposition based optimization method is applied to the equivalent mixed-integer linear programming problem (MILP) of the original DQAP. DP technique is introduced into the method in order to determine the sub-optimal solution and help accelerating the convergence rate. BD generates a database of a subset of feasible solutions for DP to determine an approximate optimal solution. For large-scale DQAPs, the solution of MILP problem in the master problem is further approximated by the round-up of the solution from the relaxed linear assignment problem using the original Hungarian method. The proposed method is tested with large-scale DQAPs against simulated annealing, tabu search and genetic algorithm. Starting from a random initial layout or the best solution from simulated annealing, the result from the method is comparable but is not as good as that from other methods. In order to accelerate BD, a trust-region constraint is implemented into the master problem. The convergence rate improves significantly but the cost of the solution improves slightly. A successive adaptation procedure is implemented to improve its performance. The result shows that the proposed method is competitive with other methods.

For implementing the logic-based model to the DQAP, the proposed methodology provides efficient solutions for small-scale problems. It takes considerable amount of computing time and provides poor solutions when the number of candidates is getting larger.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

It is my great fortune to have Associate Professor Peerayuth Charnsethikul as my advisor and my mentor. He has continually impressed and inspired me with his intellect, insight and kindness. Without his advice, patience, encouragement, sense of humor and his belief in my potential, many things would not have been possible in the past seven years of my study. I am sure that I could not have chosen a better advisor.

I also wish to express my sincere gratitude to Associate Professor Vira Chankong, Associate Professor Prapaisri Sudasana Na Ayudthaya, Associate Professor Saeree Svetaseani and Associate Professor Santi Wiriyawit for serving as my committee and most importantly their contributions in different ways during the course of study. Likewise, I would like to thank to Assistant Professor Jittat Fakcharoenphol and Dr. Peerapong Triyacharoen, graduate school representatives during my qualifying examination and final examination respectively.

I would like to gratefully acknowledge the Faculty of Engineering at Si Racha, Kasetsart University for partial financial support during my study.

I thank to the Dean, Associate Professor Kiatiyuth Kveeyarn, the former Dean, Assistant Professor Oueichai Chirachon, colleagues at Faculty of Engineering at Si Racha and tons of friends in Faculty of Engineering (Bangkhen) for their supports and cheerful encouragement.

Finally I must thank my family especially my dearest mom for their perpetual support and encouragement in every way. Their unconditional love and care have comforted me during this long and difficult study. I am also grateful to my VJ who has serendipitously shown up in my life and become my personal savior. I am sure they are even more pleased than I am to see this thesis complete.

Sirirat Muenvanichakul

February 2009

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS AND SYMBOLS

i	index for facilities
j	index for locations
k	index for facilities
l	index for locations
t	index for time periods
X_{ij}	equal to 1 if facility i is assigned to location j
X_{ijt}	equal to 1 if facility i is assigned to location j at period t
Y_{ijklt}	equal to 1 if facility i is assigned to location j and facility k is assigned to location l at period t
$M_{ijl(t+1)}$	equal to 1 if facility i is assigned to location j and reassigned to location l at period t+1
Y_{it}	location which facility i assigned to (at period t)
f_{ik}	flow between facility i and facility k
d_{jl}	distance between location j and location l
C_{ijkl}	assignment cost of assigning facility i to location j and facility k to location l
C_{ijklt}	assignment cost of assigning facility i to location j and facility k to location l at period t
R_{ijlt}	rearrangement cost of assigning facility i to location j at period t and change to location l at period t+1
F_{ikt}	flow between facility i and facility k at period t
$D_{Y_{it}Y_{kt}}$	distance between location Y_{it} and location Y_{kt} at period t
$R_{Y_{it}Y_{i(t+1)}}$	rearrangement cost of assigning facility i to location j at period t and change to location l at period t+1
N, n	numbers of facility/location
T	numbers of time period

BENDERS' DECOMPOSITION BASED HEURISTICS FOR THE DYNAMIC QUADRATIC ASSIGNMENT PROBLEM

INTRODUCTION

Facility layout analysis is necessary not only for the design of new facilities, but also for the redesign of existing facilities due to the introduction of new products, the installation of new equipment or processes, or the realization of an increase or decrease in throughput volume. The dynamic facility layout problem (DFLP) is one in which the layout arrangement of a facility (i.e., the relative location of departments) is determined for each period of a finite planning horizon. The costs associated with this model are the material handling costs for each period in the planning horizon, typically based on the well-known quadratic assignment problem (QAP) formulation, and any rearrangement costs involved in changing the layout between periods. The rearrangement costs may consist of fixed costs – in which a given cost is incurred whenever a rearrangement is made, independent of the departments that are affected – or variable costs depending only on those departments being moved. In this research the situation with only variable rearrangement costs is focused. Anyway, the DFLP can be called as the Dynamic Quadratic Assignment Problem (DQAP). In the dynamic model, even symmetric layouts must be considered, since different variable rearrangement costs will result from the different layouts. Thus, the total number of possible solutions to the general dynamic facility layout problem is $(n!)^T$, where n is numbers of facilities and locations and T is numbers of discrete time period. Notice that the DQAP is a very hard problem from theoretical (and also the practical) of point of views. Not only can the DQAP not be solved efficiently, but it cannot even be an approximated efficiently within some constant approximation ratio because of very high computation time requirements.

A first attempt to solve the DQAP would be to eliminate the quadratic term in the objective function in order to transform the problem into a (mixed) 0-1 linear programming (MILP). The linearization of the objective function is usually achieved by introducing new variables and new linear constraints. However, the very large

number of new variables and constraints poses an obstacle for efficiently solving the resulting linear integer programs. Clearly, the optimal solution of the continuous relaxation of the MILP formulation is a lower bound for the optimal value of the corresponding DQAP. Moreover, each feasible solution of the dual of this relaxation is also a lower bound.

Since DQAP is a hard problem from its theoretical standpoint, no efficient algorithm is known for this problem. All existing algorithms are in principle enumeration methods like cutting-plane algorithms. Traditional cutting-plane algorithms for QAP have been developed by different authors, for example, Bazaraa and Sherali (1982). This algorithm makes use of MILP formulations for the QAP that are suitable for Benders' decomposition.

Decomposition methods are mainly used to solve iteratively large-scale optimization problems and they have naturally been designed initially for structured linear programs. Benders (1962) has considered linear programs with complicating discrete variables, called mixed-integer programs. The main idea behind this approach is to project the original problem onto the space of integer variables in order to derive an equivalent master problem, and to solve this complicating problem via a relaxation by generating bender feasibility and optimality cuts as prompted by the subproblems that are obtained by fixing the integer variables. Unfortunately, the master problem can be seen as an integer problem with a large number of implicit constraints and the time required for this method to converge is too large, hence it may solve to optimality for only very small problems.

Due to the poor convergence rate of Benders' decomposition, it is highly skeptic for large-scale application unless some accelerating techniques are introduced into the method. A trust-region approach consists of introducing a trust-region constraint in the master problem that bounds region where the problem is trusted to present well the original constrained problem.

In additions, the concept of approximate dynamic programming is applied to the DQAP to find an upper bound for the general problem that dominates all previous bounds. It is modified from the idea of Rosenblatt (1986) but not all possible layouts can be determined. The main idea is to take the advantage of benders cuts producing a set of feasible solutions and then to find an optimal solution to the problem at exceptionally reduced solution times.

This thesis is focusing on the development of Benders' decomposition based method for the DQAP. Following Hooker(2000), the equivalence of the DQAP to a MILP problem with certain additional constraints is demonstrated. Benders' decomposition is implemented to the MILP problem. The performance of proposed method is enhanced by means of approximate dynamic program. Approximate MILP solution and trust-region constraints are also introduced into the master problem of Benders' decomposition to further improve the convergence rate of the proposed method in large-scale problems.

OBJECTIVES

The objective of this research is to develop a set of efficient methods for solving DQAP under the following scope.

1. Equal numbers of facilities and locations.
2. DQAP problem statement assumption.

Therefore, steps to achieve the objectives are as follows:

1. To formulate the mathematical model of the DQAP
2. To develop Benders' decomposition as an optimization based heuristic for solving the DQAP
3. To implement the proposed algorithms and conduct computational experiments and verify their efficiency and effectiveness.

LITERATURE REVIEW

A review of the facility layout problem by Meller (1995) mentioned the categories of models and algorithms and the objectives based on the ways to measure the distance (Rectilinear metric and Euclidean metric) and the variety assumption of the objective for finding an optimal solution. The most of the algorithms use a distance-based objective to measure the layout efficiency. The others are adjacency-based objective and α-weighted criteria.

Furthermore, the facility layout problem can be represented with other models as graph-theoretic (Meller 1995). The area of the departments is ignored. A node represents each department and a relationship connecting between two adjacent departments is represented by arc. The objective function is to maximize the flow (positive flow) between all department pairs. Anyway, this review has been focused on QAP approaches and the recent available algorithms.

A survey about the dynamic facility layout problem (DFLP) has been published by Balakrishnan and Cheng (1998) that explained the state of the research on the DFLP. They gave about explanations about algorithms adapted for DFLP along with their comparisons. There are many approaches on the DFLP, which will be discussed next.

In the fundamental paper on the DFLP, Rosenblatt (1986) developed an optimal solution methodology using dynamic programming (DP) model focused on the situation with only variable rearrangement cost. The recursive relationship was established as follows:

$$C^*_{tm} = \min_k \left\{ C^*_{t-1,k} + R_{km} \right\} + F_t^m$$

and $C^*_{0k} = 0, \qquad \forall k$.

where

R_{km} = the arrangement cost from layout k to layout m

F_t^m = the material handling cost for layout m in period t

C_{tm}^{*} = the minimum cumulative total cost of using layout *m* at period *t*

In DP, a stage will correspond to a period and a state will correspond to a specific layout. The maximum number of different layouts that need to be considered is $(n!)^{T}$, where n is the number of facilities and, locations and T is the number of considering time periods. The total number of layout combinations in each period results in a very large problem means there are $n!$ states in each of the T stages. Therefore, the objective of DP is to trade off between the material handling cost within the facility and the rearrangement costs for the departments that may need to be relocated within the facility. Rosenblatt's dynamic programming model, to restrict the state space of the model, it was determined that any layout arrangement *m* for a given period does not need to be considered if the difference between the total cost arrangement, F_t^m, and the cost of the optimal static solution for that period, F_t^*, is greater than the difference between the total cost values of the upper bound, C^+, and the lower bound, C^-, that is,

$$F_t^m - F_t^* \geq C^+ - C^-$$

Thus, only the set of static solutions violating the above constraint period need to be considered.

To generate an upper bound value is to continue with the best incumbent feasible solution to the multi-period problem (including rearrangement cost). In this case, a lower bound is the sum of the minimum costs of SFLP in each period over the entire periods without the rearrangement cost, $C^- = \sum_t F_t^*$. In this study, DP can be used to solve DFLP in an optimal or a heuristic manner that based on the computational efficiency with which SFLP can be solved. He recommended that for an effective solution in large problems, DP could be used with smaller number of states be considered.

Luangpaiboon (1995) proposed using DP approach presented by Rosenblatt to test small problems in size 5 and 6 departments over 1, 5 and 8 time periods

respectively in both cases of complete searching space and restricted state space. DP is efficient on the small problem taken less computational time.

In fact, for many practical situations, the primary cost associated with the rearrangement cost of a facility is the fixed cost that results from the disruption, or possible shutdown, of its operations (Urban, 1998). This paper illustrates the DFLP where the rearrangement costs are assumed to be fixed regardless of the departments rearranged over time. The objective function of this scenario reflects the simple tradeoff between the workflow costs and the rearrangement costs:

$$TC = \sum_{t=1}^{T}\left[\sum_{i}\sum_{j}\sum_{k}\sum_{l} f_{ikt} d_{jl} X_{ijt} X_{klt} + r_t Z_t\right]$$

where f_{ikt} is the unit workflow costs

r_t is the unit rearrangement costs

The incomplete dynamic programming algorithm is proposed to this problem. The decision in each period depend on whether rearrangement be made in a given period vector $Z = (z_2,\ldots, z_T)$. The static QAP can then be solved for each sequence that begins with either period 1 or a period p, where $z_p = 1$ and ends with either period T (the end of the planning horizon) or the period before a period q, where $z_q = 1$.The static QAP will need to be solved m+1 times where m is the consecutive number of ones in the vector. The value of the total workflow used in each sequence, or "subproblem", will be the sum of the workflows for all periods in that sequence. It is then QAP sub-problem as follows:

$$\text{Minimize} \quad F_{p,q-1} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\left[\sum_{t=p}^{q-1} f_{ikt}\right] d_{jl} X_{ij} X_{kl}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} X_{ij} = 1 \qquad ,\forall j$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad ,\forall i$$

$$X_{ij} \in \{0,1\} \qquad \forall i, j$$

To determine the optimal solution to the overall problem, this procedure will be performed for every possible z vector. So, solving DFLP will require the solution of $T(T+1)/2$ QAP sub-problems and then the solutions form the arcs on a shortest path formulation with each period represented by a node. This study considers the problem in size of 9 departments and 6 periods due to assigned range of fixed rearrangement cost. Additionally, the another aspect of the research are to develop as stronger lower bound for the general DFLP with a minimal increase computational requirements and to identify the stronger upper bound to reduce the total computational effort required solving the DFLP. Thus, the number of states required in DP for the general DFLP is dependent on tight upper and lower bound values.

Charnsethikul (1999) proposed an exact algorithm for DQAP based on a branch and bound technique. The lower bound is proposed to eliminate unimproved solutions, and the technique doesn't need all possible assignments in each period as the dynamic programming need. He also recommended that the lower bound quality should be further investigated.

Urban (1993) has presented a heuristic for solving DFLP without employing the optimal solution of QAP. The algorithm uses the steepest–descent pair wise exchange procedure, which is similar to the idea of computerized relative allocation of facilities technique (CRAFT). The procedure starts with initiating an initial layout assignment along considered periods. The difference is that rearrangement costs are included. The rearrangement costs may consist of an associated cost, which the particular department has been moved, or fixed cost, which rearrangement is made for any period whichever departments are moved. The heuristic is proposed by makes use of forecast window; *m* to find difference sets of good layout for the given period. The forecast window is the number of periods when pairwise exchange is performed, and range from one to number of periods, *T*. Figure 1 illustrates pair wise exchange when forecast window $m = 2$. It shows that the material flow in period 1 and period 2 are determined for the layout in period 1. Similarly, the considered cost in period 2 and 3 are combined to determine the layout in period 2. The number of possible pairs applying pairwise exchange are $n(n-1)/2$. The pair of departments will be

interchanged at each iteration to maximize the reduction in total cost. That is finally expected to do well in the solutions.

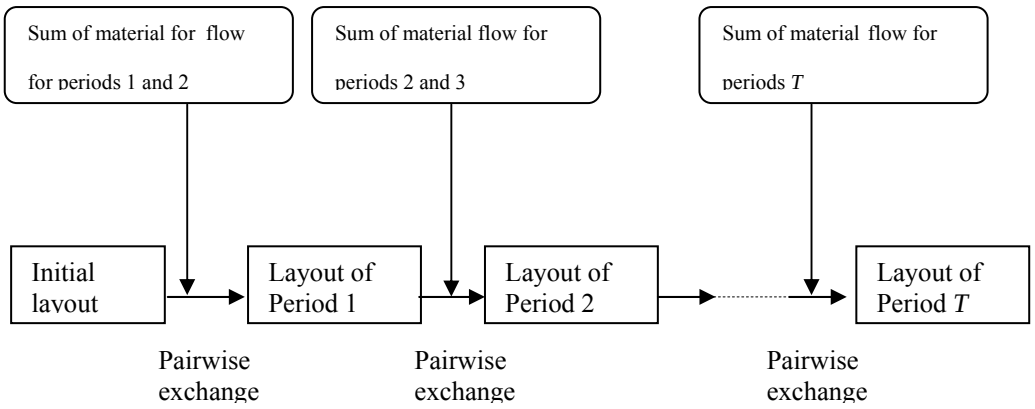


Figure 1 Pairwise exchange when forecast window $m = 2$

Balakrishnan *et al*. (2000) presented two improvements for Urban's heuristic (1993). The first approach is to reduce the one disadvantage of Urban's procedure by adding a backward method into the procedure. The backward pass pairwise is performed after each forecast window receives its solution. It performs on each of solutions, and starts in period (t-1) and goes backward until the first period of the planning. A considerable different between backward and forward pass methods is considered rearrangement cost. In a backward pass, the rearrangement cost between period (t-1) and t, and the rearrangement cost between period t and (t+1) are considered, while, in a forward pass, the rearrangement cost between period (t-1) and t are only considered. The last approach is to combine Urban's heuristic and dynamic programming, DP since the layouts searching by Urban tend to be good. DP is performed to eliminate duplicated layouts generated along the entire windows. For computational results, there are improvements on the results compared to Urban's procedure for almost every case.

Conway and Vekataramanan (1994) solved the constrained DFLP by using genetic algorithm with only crossover component (CONGA). In their procedure, a string in the population consists of ($n \times t$) digits. Two strong strings are selected from the total number of population (n!) based on the fitness function for crossover operator.

Then, a random splicing position is generated and the strings are split and swapped. The string with lower cost is allowed to survive to the next generation.

Base upon genetic algorithm (GA) characteristic previously described, Balakrishnan and Cheng (2000) and Muenvanichakul (2000) presented the different GA aspect from the existing implementation.

Balakrishnan and Cheng (2000) applied GA algorithm in three different ways. First, they adopt a different crossover operator to increase the search space. Then a point-to-point crossover operator is used to cut them at every position, starting from the first position to the last position in the string. A feasibility test is applied each time to eliminate illegal child layouts. Second, they applied mutation to increase population diversity. It may apply in the illegal child layout with minimum cost. And third, they use a new replacement strategy to increase population diversity. In addition, the special characteristic procedure uses a GA with nested loops (NLGA). The inner-loop uses a steady-state replacement approach and replaces the most unfit individual in each generation. The outer loop will replace a large number of unlucky individuals generation to ensure that inner loops work with different populations. Thus, this enlarges the search space and should lead to better solutions.

An improved GA for DFLP is developed and tested in Muenvanichakul's research. The TBX1 crossover operator and the scramble mutation operator are used for improving the local search space. The outstanding consequence of GA is a development of efficient rules and unbiased search over a limited area. Therefore, the fitness function in every string of population is evaluated and then, the roulette wheel is created.

Kochhar and Heragu (1999) have explored the design of a multiple-floor dynamic facility, which can respond effectively to the change in product demand and mix change in a continuously evolving work environment. To demonstrate the dynamic heuristically operated placement evolution; called DHOPE, for solving the problem. DHOPE is an extension of the genetic algorithm-based heuristic. It is

concerned with the design of dynamic facility over two consecutive planning period. It is interested in developing a near-optimal layout for the second period given a layout for the first, DHOPE attempts to find a layout that minimizes material flow cost (during the second period) as well as machine rearrangement costs.

Tabu search (TS) method is one of the well-known metaheuristic initially proposed by Glover et al. The pairwise interchange is used to evaluate candidate move in a local neighborhood search. The general concept of this method is to develop a system for maintaining records of local optimum solutions during a select search scheme in order to avoid the search pattern to repeat convergence to the previous obtained solution called short term memory. The tabu-list is an important tool in guiding the search direction, given the determination of an effective set of attributes for defining tabu status. The best non-tabu move is implemented if it result in a better solution. Long term memory is the frequency move of "move" from every recorded value occurred in the short-term memory to be considered as the penalty adjustment in the restart of searching procedure. Finally, if there is an excellent solutions never happened in last evaluation of such prohibited "tabu move", it can be expected and accepted the solution. This case is called aspiration criterion applied along the computational time (Muenvanichakul, 2000).

Balakrishnan (1998) explained tabu search procedure for DFLP proposed by Kaku and Mazzola. The algorithm has two searching aspects as diversification and intensification strategies. Diversification strategies are used to ensure that different regions of the research space are explored for better solutions and intensification strategies are the procedure to do more searches in a neighborhood such as reducing the tabu list.

Baykasoğlu *et al*. (2001) developed simulated annealing (SA) based procedure for DFLP and reported resulted for the test of problem. In the test SA found the optimal solutions and performed better than the DP of Rosenblatt and the GA of Conway and Vekataramanan. In case of computational time, SA performed considerable well than Balakrishnan and Cheng's GA.

Luangpaiboon (1995) developed a simulated annealing approach for solving the DFLP in large problem with size 20 departments over 5 periods and 40 departments over 3 periods. CRAFT was used to generate static solutions and the other was that instead of CRAFT, used random layouts be starting solutions. However, in case of small problem, the solutions from SA, CRAFT and DP were compared together.

Urban (1998) illustrated two separate heuristics for DFLP with fixed rearrangement costs as a greedy randomized adaptive search procedure (GRASP), and an initialized multi-greedy algorithm. GRASP is an iterative search procedure that has been successfully applied in a variety of combinatorial optimization problems. Each iteration of GRASP consists of two phases: an adaptive construction procedure and a local search procedure, a feasible solution is randomly generated in construction phase and then a local search is conducted to identify improved solutions. For DFLP with fixed rearrangement costs, GRASP will be independently conducted for each of the $T(T+1)/2$ sub-problems, developed by utilizing incomplete dynamic programming, to be used in the shortest-path network. To use the solution to each of these sub-problems be an initial solution to the problem under consideration. The concept of the initialized multi-greedy is similar to GRASP. The different is to allow appropriate information be transferred between sub-problems. In local search, all the two period sub-problems are solved using the two appropriate single-period solutions and k-2 randomly generated the solutions, repeat until $T(T+1)/2$ sub-problems are solved. In this case, k is numbers of department considered.

Lacksonen and Enscore (1993) conducted tests on various static layout algorithms to handle DFLP. The five algorithms are selected as CRAFT, cutting planes, branch and bound, dynamic algorithm and cut trees.

The proposed TS method and GA algorithm are applied to test the problem from Luangpaiboon's study in two aspects. First, the research has studied separately the efficiency and effectiveness of procedure in case of various changes of their parameters. Second, to combine these methods is given higher performance and

efficiency for searching a solution. The process begins with searching a solution either within limited time or the end of evaluation by either procedure and then converts the solution to an initial value for another procedure. This is switched among TS, GA and SA in both small and large problems (Muenvanichakul, 2000).

MATERIALS AND METHOD

Materials

In this research, the following materials are used to conduct the numerical experiments testing the propsed algorithms for DQAP are

1. A laptop with 1.66 GHz. Inter® Core 2 CPU., 1 GB of Ram, Microsoft O.S.

2. Software: MATLAB version R2007b

Methods

The methodology used to develop the algorithms for solving the DQAP are Benders' decomposition and Dynamic Programming.

1. Problem Statement

1.1 Quadratic Assignment Problem (Cela (1998))

QAP is a classical problem in location theory, in which the cost associated with placing n facilities to an equal number of sites. The objective is to minimize the sum of assignment cost for an entire related pairs. For each pair of facilities (i, k) a flow of communication; f_{ik} is known, and for each pair of sites (j, l) the corresponding distance; d_{jl} is known. The assignment cost between facilities i and k, given that i is located at site j and k is located at site l, is $f_{ik} * d_{jl}$. Furthermore, the problem deals with a finite number of facilities and sites, and one facility is to be assigned to each site. The site, for example, might be rooms in a plant, and the facilities might be departments to be assigned to the rooms. Alternatively, the site might be actual geographic sites, and the facility might be plants to be assigned to the sites.

The first published statement of the QAP was introduced in 1957 by Koopmans and Beckman as follows:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik} d_{jl} X_{ij} X_{kl} \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} X_{ij} = 1 \qquad ,\forall j$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad ,\forall i$$

$$X_{ij} \in \{0,1\} \qquad ,\forall i, j$$

where

X_{ij} = 1 if facility i is assigned to site j

= 0 otherwise

n = number of facilities or sites

$f_{ik} * d_{jl} = C_{ijkl}$ if facility i is assigned to site j and facility k is assigned to site l

Notice that the first two constraints ensure that exactly one facility is assigned to each site and each facility is assigned to exactly one site, respectively.

QAP is similar in structure to the classical linear assignment problem (LAP). If there is no interaction among facilities (facility locations are independent of each other), setting $f_{ik} = 0, \forall i, k.$, therefore, the model becomes the LAP:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} X_{ij} \tag{2}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} X_{ij} = 1 \qquad ,\forall j$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad ,\forall i$$

$$X_{ij} \in \{0,1\} \qquad ,\forall i, j$$

where

X_{ij} = 1 if facility i is assigned to site j

= 0 otherwise

n = number of facilities or sites

C_{ij} is the cost of assigning facility i to site j

Additionally, LAP is a special case of the well-known transportation problem that the facility might represent a set of sources, and the site might represent a set of destinations. For each source, the amount of supply exactly equal one, and for each destination, the amount of demand is also equal to one. The cost of transporting is C_{ij}.

1.2 Dynamic Quadratic Assignment Problem

A problem in the dynamic facility layout planning (DFLP) is a decision problem of finding the optimal location assignments among a set of facilities over a set of discrete periods. During this time, many of the parameters of the problem such as demands and distribution costs are likely to change. The objective is to minimize the sum of flow costs and rearrangement cost over all discrete time periods. The application of this problem is necessary not only for the design of new facilities, but also for the redesign of existing facilities due to introduction of new products, the installation of new equipments or processes, or realization of an increase or decrease in throughput volume.

DFLP is therefore generally formulated as a modified QAP; of assigning equal area facilities to discrete locations. The mathematical model is as follows:

Minimize
$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt} X_{ijt} X_{klt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt} X_{ijt} X_{il(t+1)} \quad (3)$$

Subject to
$$\sum_{i=1}^{n} X_{ijt} = 1 \quad , j = 1, \ldots, n \ , t = 1, \ldots T \quad (4)$$

$$\sum_{j=1}^{n} X_{ijt} = 1 \quad , i = 1, \ldots, n \ , t = 1, \ldots T \quad (5)$$

$$X_{ijt} \in \{0,1\} \quad , \forall i, j, t \quad (6)$$

where

X_{ijt} = 1 if department *i* is assigned to location *l* at period *t*

= 0 otherwise

n = number of departments or locations

T = number of time periods

The cost of assigning, which facility *i* to site *j* and facility *k* to site *l* at period *t*, is $C_{ijklt} = F_{ikt} * D_{jlt}$. There is also additional cost, called Rearrangement Cost; R_{ijlt}, which the facility *i* is located on site *j* at period *t*; X_{ijt}, and moved to site *l* at period (*t*+1); $X_{il(t+1)}$.

In order to directly search for a solution of DFLP, a method in solving SFLP is modified by minimizing the total of transportation costs or the material handling cost between the given number of facilities for all given periods plus rearrangement costs for a series of static layout decisions. The rearrangement cost may consist of fixed cost whenever the rearrangement is made no matter what departments are affected or variable costs depending on those departments being moved.

1.3 Linearization of QAP

It is well known that QAP is strongly NP-hard that is quite difficult to deal with in case of an optimal solution. Linearization is, one of many ideas to cope with the QAP by transforming a quadratic equation in QAP objective into a set of linear inequalities and a corresponding linear equation. Equivalent formulation of the problem is as a mixed integer linear programming (MILP). Several researchers have stated different linearization models as follows.

The non-linearity of the objective is linearized by new variable Y_{ijkl} (Hooker, 2000). So, the model becomes:

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik} d_{kl} Y_{ijkl}$

Subject to

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad ,\forall\, j$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad ,\forall\, i$$

$$Y_{ijkl} \geq X_{ik} + X_{jl} - 1 \qquad ,\forall\, i, j, k, l$$

$$X_{ij} \in \{0,1\} \qquad ,\forall\, i, j$$

$$Y_{ijkl} \geq 0 \qquad ,\forall\, i, j, k, l$$

1.4 Linearization of DQAP

The linearization of QAP can be modified for the DQAP by introducing two new variables. The variable Y_{ijklt} is equally one if the facility *i* is assigned to site *j* and facility *k* is assigned to site *l*. Another variable is $V_{ijl(t+1)}$, which represents the facility *i* locate to site *j* at period *t* and relocate to site *l* at period *t*+1. The model therefore becomes:

Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt} Y_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt} M_{ijl(t+1)} \qquad (7)$$

Subject to

$$Y_{ijklt} \geq X_{ijt} + X_{klt} - 1 \qquad ,i = 1,..,n, j = 1,\ldots, n \;, t = 1,\ldots T \qquad (8)$$

$$M_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1 \qquad ,i = 1,..,n, j = 1,\ldots, n \;, t = 1,\ldots T \qquad (9)$$

$$\sum_{i=1}^{n} X_{ijt} = 1 \qquad ,j = 1, \ldots, n \;, t = 1,\ldots T \qquad (10)$$

$$\sum_{j=1}^{n} X_{ijt} = 1 \qquad ,i = 1,\ldots, n \;, t = 1,\ldots T \qquad (11)$$

$$Y_{ijklt}, M_{ijl(t+1)} \geq 0 \qquad (12)$$

$$X_{ijt} \in \{0,1\} \qquad ,\forall\, i, j, t \qquad (13)$$

Extending the theorem and proof in Lawler (1963), it is possible to demonstrate that the linearization of DQAP is equivalent to DQAP. Let the DQAP defined in (3) to (6) be designated problem *Q*, and the MILP defined in (7) to (13) be designated problem *L*. The following theorem assures the equivalence of *Q* and *L* for any given set of cost coefficients.

Theorem 1: The feasible solutions of problems *Q* and *L* can be placed in one-to-one correspondence with equal values of the cost functions. A feasible solution $X^{(Q)}$ of *Q* corresponds to a feasible solution $(X^{(L)},Y,V)$ of *L* if and only if $X^{(Q)}=X^{(L)}$.

Proof: It is sufficient to show that the constraints of problem *L* are such that for any given permutation matrix $X^{(L)}$ at a given period *t*, *Y* at period *t* and *V* from period *t* to *t+1* are determined uniquely by the relations

$$Y_{ijklt} = X_{ijt}X_{klt} \text{ , and}$$

$$M_{ijl(t+1)} = X_{ijt}X_{il(t+1)}.$$

Since all of the variables are restricted to the value of 0 and 1, these relations are equivalent to

$$Y_{ijklt} = 1 \Leftrightarrow X_{ijt} = X_{klt} = 1, \text{ and}$$

$$M_{ijl(t+1)} = 1 \Leftrightarrow X_{ijt} = X_{il(t+1)} = 1.$$

It follows immediately from

$$Y_{ijklt} \geq X_{ijt} + X_{klt} - 1 \text{ that } Y_{ijklt} = 1 \Rightarrow X_{ijt} = X_{klt} = 1, \text{ and}$$

$$M_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1 \text{ that } M_{ijl(t+1)} = 1 \Rightarrow X_{ijt} = X_{il(t+1)} = 1.$$

In order to prove the converse, let $X_{ijt} = X_{klt} = 1$. Then, from the constraints

$$Y_{ijklt} \geq X_{ijt} + X_{klt} - 1$$

$$M_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1,$$

It follows that

$$Y_{ijklt} \geq 1 \qquad (14)$$

$$M_{ijl(t+1)} \geq 1 \qquad (15)$$

Since the objective function is to minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt} Y_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt} M_{ijl(t+1)}$$

and $C_{ijklt} \geq 0$ and $R_{ijlt} \geq 0$ by definition, Y_{ijklt} and $M_{ijl(t+1)}$ must choose the minimum feasible values accordingly to (14) and (15). Therefore, it follows that

$$Y_{ijklt} = 1$$

$$M_{ijl(t+1)} = 1$$

whenever $X_{ijt} = X_{klt} = 1$ Q.E.D.

2. Algorithms Development

2.1 Benders' decomposition

For solving large scale mixed integer programming problems, Benders' decomposition technique, which is presented by J. F. Benders (1962), is one of possible approaches. The algorithm solves mixed integer programming via structure exploitation by decomposing a mixed integer problem into two problems, which are solved iteratively, an integer master problem and a linear sub-problem. A decomposable mixed integer linear programming problem, called P1, is:

P1: Minimize $f^T x + c^T z$ (16)

Subject to $Gx \geq b_1$ (17)

$Hx + Az \geq b_2$ (18)

$Dz \geq b_3$ (19)

x is integer, $z \geq 0$

where x = integer variable and z = continuous variable.

and (16) represents the minimize objective function of the MILP

(17) represents the integer constraint

(18) represents the mixed-integer constraint

(19) represents the continuous constraint

To develop the decomposition of this problem, the procedure is proceeded by iteratively fixing feasible points *x** and solving the sub-problem to yield values of its dual variables. The corresponding primal and dual models can be written as:

Primal: Minimize $c^T z$

Subject to $Az \geq b_2 - Hx$ (α -dual variable)

$Dz \geq b_3$ (γ - dual variable)

$z \geq 0$

Dual: Maximize $(b_2 - Hx)^T \alpha + b_3^T \gamma$

Subject to $A^T \alpha + D^T \gamma \geq c^T$

$D^T \gamma \leq 0$

$\alpha, \gamma \geq 0$

According to the duality theorem of linear programming, a master problem, MP2 is formed by including the dual information to generate a better guess X.

MP2: Minimize $f^T x + q$

Subject to $q \geq (b_2 - Hx)\alpha + b_3 \gamma$

$Gx \geq b_1$

x is integer

The objective function of MP2 is similar to P1 since q is an approximation of $c^T z$. Each iteration, a new constraint is added to the master problem to strengthen the lower bound in identifying % optimality of the obtained solution.

Benders' decomposition can be summarized step-by-step for MATLAB as in Figure 2.

```
% Benders' decomposition for Dynamic Quadratic Assignment Problem
% Input: Cijklt, Rijltp, Aeq, beq, xijt0, Ncut_max
% Output: xijt, total_cost

% Initialization
xijt = xijt0                    % layout
LB = -Inf; UB = Inf             % bounds
A_master = []; b_master = []    % allocate constraint for master problem

% Iteration (over cuts)
FOR iter = 1, Ncut_max
        % Solve sub-problem
        CALL benders_sub        % input: xijt0, Cijklt, Rijltp
                                % output: Uijklt, Vijltp, z_sub
        % Check UB
        IF z_sub < UB
                UB = z_sub      % z_sub is the cost of sub-problem
        ENDIF
        % Solve master-problem
        CALL benders_master     % input: Uijklt, Vijltp, A_master, b_master, Aeq, beq
                                % output: xijt, z_master, A_master,b_master
        % Check LB
        IF z_master > LB
                LB = z_master           % z_master is the cost of master-problem
        ENDIF
        % Convergence Test
        IF LB > mean(z_sub) -3 sigma(z_sub)
                % Display and save the result
                WRITE & SAVE xijt, total_cost
                RETURN
        ENDIF
END

IF DTC > Tol                    % if the difference of total cost > the given tolerance
        % Display warning
        WRITE warning message – iter_max reached, no converged solution
        % Display and save the result
        WRITE & SAVE xijt, total_cost
END
```

Figure 2 Algorithm of Benders' decomposition for DQAP

Since the linearization DQAP is a mixed integer problem, Benders' decomposition can be used to decompose the model into its integer and continuous parts. Consider the primal problem:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt} Y_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt} M_{ijl(t+1)} \tag{20}$$

$$\text{Subject to} \quad Y_{ijklt} \geq X_{ijt} + X_{klt} - 1 \qquad ,i = 1,..,n, j = 1,\ldots, n \ , t = 1,\ldots T \tag{21}$$

$$M_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1 \quad ,i = 1,..,n, j = 1,\ldots, n \ , t = 1,\ldots T \qquad (22)$$

$$\sum_{i=1}^{n} X_{ijt} = 1 \quad ,j = 1, \ldots, n \ , t = 1,\ldots T \qquad (23)$$

$$\sum_{j=1}^{n} X_{ijt} = 1 \quad ,i = 1,\ldots, n \ , t = 1,\ldots T \qquad (24)$$

$$Y_{ijklt}, M_{ijl(t+1)} \geq 0, \qquad X_{ijt} \in \{0,1\} \quad ,\forall \ i, j, t$$

Referring to the above details, the equivalent of original model and the DQAP are as follows.

The Original model	The DQAP model
Z	Y_{ijklt} , $M_{ijl(t+1)}$
X	X_{ijt}
$c^T z$	$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} C_{ijklt} Y_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} R_{ijlt} M_{ijl(t+1)}$
$Gx \geq b_1$	$\sum_{i=1}^{n} X_{ijt} = 1 \quad ,j = 1, \ldots, n \ , t = 1,\ldots T$
	$\sum_{j=1}^{n} X_{ijt} = 1 \quad ,i = 1,\ldots, n \ , t = 1,\ldots T$
$Hx + Az \geq b_2$	$Y_{ijklt} \geq X_{ijt} + X_{klt} - 1 \quad ,i = 1,..,n, j = 1,\ldots, n \ , t = 1,\ldots T$
	$M_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1 \quad ,i = 1,..,n, j = 1,\ldots, n \ , t = 1,\ldots T$

Then, the corresponding dual problem is

Maximize

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T} (X_{ijt} + X_{klt} - 1) U_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1} (X_{ijt} + X_{il(t+1)} - 1) V_{ijl(t+1)}$$

Subject to

$$0 \leq U_{ijklt} \leq C_{ijklt} \quad ,i = 1,..,n, j = 1,\ldots, n \ , t = 1,\ldots T$$

$$0 \leq V_{ijl(t+1)} \leq R_{ijlt} \quad ,i = 1,..,n, j = 1,\ldots, n \ , t = 1,\ldots T$$

$$X_{ijt} \in \{0,1\} \quad ,\forall \ i, j, t$$

For the primal model, the variables Y_{ijklt} is possibly greater than or equal one or zero or minus one according to the value of X_{ijt}. Since C_{ijklt} and R_{ijlt} are always greater than or equal to 0, an optimal solution; Y^*_{ijklt} will be equal to 1 if $Y_{ijklt} \geq 1$, and 0 if $Y_{ijklt} \geq 0$ and familiarly, for the optimal value of variable M^*_{ijlt}. Consequently, the objective value of the dual model depends on what value of $\left(X_{ijt} + X_{klt} - 1\right)$ is. However, there are exactly three alternative solutions as –1, 0 and 1. Therefore, there are sets of variables Y_{ijklt} and $M_{ijl(t+1)}$ may take the value (-1) and they will be automatically eliminated due to the integrality constraints. The previous summary is assured by the following theorem.

Theorem 2: There is no feasibility cut in Benders' decomposition of the linearization DQAP.

Proof: It is sufficient to show that given X_{ijt}, X_{klt} for the corresponding dual problem, to maximize the objective function

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T}(X_{ijt} + X_{klt} - 1)U_{ijklt} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1}(X_{ijt} + X_{il(t+1)} - 1)V_{ijl(t+1)}\,.$$

It follows that $(X_{ijt} + X_{klt} - 1)$ and $(X_{ijt} + X_{il(t+1)} - 1)$ must be equal to 1 and $C_{ijklt} \geq 0$ and $R_{ijlt} \geq 0$ by definition. Therefore, the optimal solutions of the dual problem will be therefore forced to the maximum value of the bound as:

$$U^*_{ijklt} = C_{ijklt}$$

$$V^*_{ijl(t+1)} = R_{ijlt}$$

Since U^*_{ijklt} and $V^*_{ijl(t+1)}$ are the solutions at extreme point, it must obtain only optimality cut to the benders master problem. Q.E.D.

The sub problem of Benders' decomposition can be summarized step-by-step for MATLAB as in Figure 3.

```
% Solver for sub-problem of Benders' decomposition for Quadratic Dynamic
%       Assignment Problem
% Input: xijt0, Cijklt, Rijltp
% Output: Uijklt, Vijltp, z_sub

% Maximize the objective function subject to bounds
Uijklt = max(0,xijt + xklt – 1)*Cijklt
Vijltp = max(0,xijt + xil(t+1) – 1)*Rijltp
```

Figure 3 Pseudo Code of Benders' decomposition – Sub (Dual) Problem

For an obtained optimal U^*_{ijklt} and $V^*_{ijl(t+1)}$, the master problem can be solved for a new solution. Iteratively, the process can be continued by adding the cutting plane for a candidate solution in order to improve local bound Z. Thus, the new adding cut is

$$Z \geq \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T}(X_{ijt}+X_{klt}-1)U^*_{ijklt}+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1}(X_{ijt}+X_{il(t+1)})V^*_{ijl(t+1)}$$

Theoretically; the objective of master problem is equivalent to the objective of the problem and the mathematic model is as follows:

Minimize Z

Subject to $Z \geq \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T}(X_{ijt}+X_{klt}-1)U^*_{ijklt}+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{T-1}(X_{ijt}+X_{il(t+1)})V^*_{ijl(t+1)}$

$$\sum_{i=1}^{n} X_{ijt}=1 \qquad ,j=1,\ldots,n \ , t=1,\ldots T$$

$$\sum_{j=1}^{n} X_{ijt}=1 \qquad ,i=1,\ldots,n \ , t=1,\ldots T$$

$$X_{ijt} \in \{0,1\} \qquad ,\forall\ i,j,t$$

The master problem can be summarized step-by-step for MATLAB as in Figure 4.

```
% Solver for master-problem of Benders' decomposition for Quadratic Dynamic
%       Assignment Problem
% Input: Uijklt, Vijltp, A_master, b_master, Aeq, beq
% Output: xijt, z_master, A_master,b_master

% Setup objective function and constraints
SETUP c_master                          % objective function for master problem
ADD constraint equation for a new cut to A_master and b_master      % add a new cut

% Approximate solution for Mixed-Integer Type Problem using SIMPLEX +
%       HUNGARIAN METHOD

% Minimize the objective function using SIMPLEX;
%       Solution space is 0 <= xijt <= 1, -∞ <= z_temp <= ∞
CALL linprog(simplex  on)               % CALL linprog function in matlab with
                                        %       simplex solver on (see help in matlab)
                                        % input: c_master, A_master, b_master, Aeq,
                                        %       beq
                                        % output: xijt, z_temp (objective function value)

% Round up layout xijt at each period to 0 and 1 using Hungarian Method.
%       The objective is to minimize the difference between the layout xijt and 1.
CALL Hungarian                  % input: (0 <= xijt <= 1)
                                        % output: xijt (= 0 or 1)

% Determine the minimum z_master that satisfies all the constraints for given
%       xijt  from Hungarian method
COMPUTE z_master
```

Figure 4 Pseudo Code of Benders' decomposition –Master Problem

2.2 Approximate Dynamic Programming

In the fundamental paper on the dynamic facility layout problem, Rosenblatt (1986) proposed a dynamic programming to develop an optimal solution methodology and identify bounding procedures. Using dynamic programming terminology, a state will correspond to a specific layout and a stage will correspond to a period. This procedure can be used to reduce the number of candidate static layouts to be examined. The effectiveness of this procedure depends on the relative magnitude of the shifting costs.

With Rosenblatt's dynamic programming model, each period in the discrete-time planning corresponds to a stage and each particular layout arrangement

corresponds to a state. Therefore, there are $n!$ states in each of the *T* stages. The total number of possible solutions is $(n!)^T$. Restricting the state space of the model was determined that any layout arrangement for a given period does not need to be considered if the difference between the total cost of the arrangement and the cost of optimal static solution for that period, is greater than the difference between the values of the upper bound and the lower bond of the model. Therefore, only the best static solutions for each period need to be considered. For a small problem, the optimal solution can be obtained but for a larger problem, using all $n!$ static layouts will result intractable problem.

Let *n* is the number of the static assigning layouts of period *t*. The main concept of this methodology is providing the optimal solution for the layouts included in its procedure. In the case that n is very large, exploring all possible solutions as n! at each period needs the capability of software and hardware to solve the problem. If $N_t < n!$, it then cannot guarantee the optimal solution for the problem since all the possible static layouts are not determined. Concerning the powerful selecting the best N_t layouts in each period, larger N_t in the set of the best ranked solutions should lead to better solutions. There are some suggestions on the method of selecting the N_t layouts where $N_t < n!$. One method is to choose them randomly, but the quality of the solutions is usually not good.

Finding the optimal solution for DQAP relies on the solution to the QAP. It is well known that static QAP is NP-complete problem, so heuristic procedures will be necessary for providing the good solutions in an efficient executing time. In this paper, the method of selecting the N_t layouts is using the best layouts which provide by the metaheuristics. Muenvanichakul (2000), genetic algorithm, tabu search method and simulated annealing method were applied to solve DQAP. Next, approximated dynamic programming, which is based on the concept of incomplete dynamic programming, is applied as an optimal solution procedure. The procedure eliminates all repeating static layouts in each period. It attempts to reduce exploring solution time and avoid constructing the repeating layouts. In approximated dynamic

programming, the n static layouts (states) in each period can be different to each other. Thus, N_t represents the number of static layouts in period (stage) t. A framework of approximated dynamic programming is presented in Figure 5.

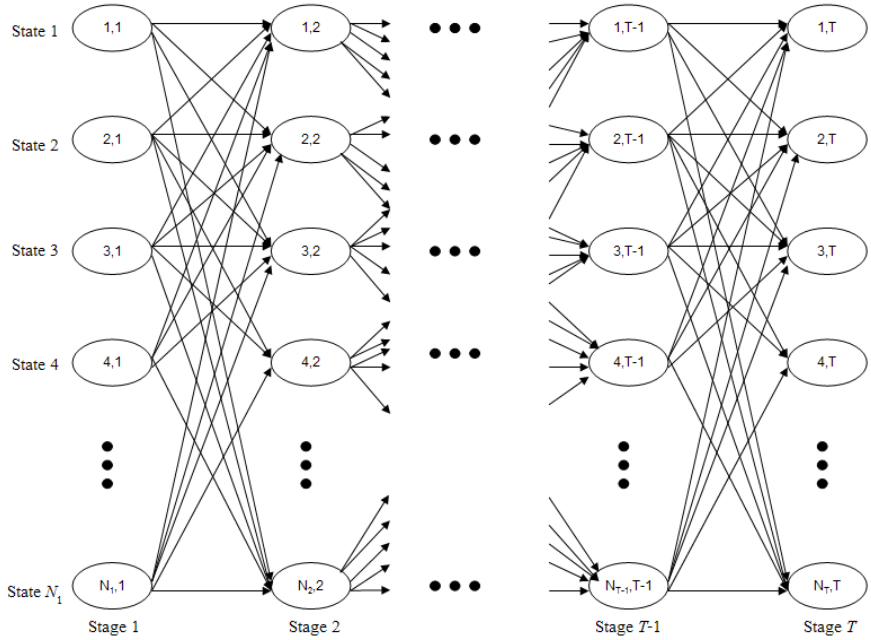


Figure 5 Approximate Dynamic Programming Framework

However, an appropriate lower bound proposed to cut off useless solutions is restricted in approximated dynamic programming because n layouts are not all possible solutions, and they are the best solutions ever found by metaheuristics. It attempts keeping the largest n. Logically, the larger n should lead the better solutions.

The dynamic programming can be summarized step-by-step for MATLAB as in Figure 6.

```
% Dynamic Programming for Quadratic Dynamic Assignment Problem
% Input: Cijklt, Rijltp, xijt, xijt_data
% Output: xijt_opt, xijt_data, TC_opt

% Layout preparation
APPEND xijt0 to xijt_data          % Append layouts from Benders' decomposition
                                   %      to layouts from previous iterations
REMOVE REPETITION xijt_data % Remove layout repetition in xijt_data

% Compute assignment cost and sort the layouts
FOR t = 1,T
        AC = AssignmentC(xijt_data,Cijklt,t)
END

[ACmin,Imin] = SORT(AC)                % ACmin = Assignment cost sort from min to max;
                                       % Imin = layout index according to ACmin

% Compute rearrangement cost of the layout corresponding to ACmin
FOR t = 1,T-1
        RC = RearrangementC(xijt_data,Imin,Rijltp,t)
END

% Compute lower bound
LB = SUM(ACmin)                    % LB = summation of minimum assignment
                                   %      cost from different periods

% Compute upper bounds
UB(1) = LB + RC                        % The layouts from minimum assignment
                                       %      cost over different periods
                                       %      +  rearrangement cost

FOR t = 1,T-1                          % The layouts for all periods are
        UB(t+1) = sum(ACmin,t)  %      the same as layout at period t;
END                                    % Only assignment cost; no rearrangement cost

% Compute different bound
DB = min(UB-LB)

% Filter layouts and generate all possible combination
FILTER OUT xijt_data               % Filter xijt_data that has assignment cost at
                                       %      each period > DB

GENARATE layout                        % Generate all combination of layout over T periods

% Find optimal layout
COMPUTE layout cost                % Compute assignment + rearrangement cost of
                                       %      all layout combination

FIND xijt_opt                          % Find layout combination that has minimum cost
```

Figure 6 Pseudo Code of Dynamic Programming

```
% Combinatorial Optimization – Benders' decomposition + Dynamic
%       Programming for Quadratic Dynamic Assignment Problem

% Read input data
READ N, T, Fikt, Djlt, Gjlt              % problem parameters
READ iter_max, Tol, Ncut_max             % optimization control parameters
READ xijt0                               % initial layout

% Setup objective function and constraints
COMPUTE Cijklt, Rijltp          % assignment and rearrangement cost matrices
COMPUTE Aeq, beq                % equality constraint i.e. sum over i = 1,
                                % sum over j = 1 without the last constraint for every period

% Determine the total cost of the initial layout
COMPUTE AC, RC, TC_old                   % assignment, rearrangement and total cost

% Iteration
FOR iter = 1, iter_max
        % Benders' decomposition for dynamic quadratic assignment
        %       problem
        CALL benders_dqap                % input: Cijklt, Rijltp, Aeq, beq, xijt0, Ncut_max
                                         % output: xijt (layouts for every cut), total_cost
        % Dynamic Programming
        CALL dp                          % input: Cijklt, Rijltp, n, T, xijt,
                                         %       xijt_data (layouts from previous iters)
                                         % output: xijt_opt (best layout),
                                         %       xijt_data (collect layouts at this iter
                                         %        to xijt_data from previous iters),
                                         %       TC_opt (best total cost from dp)

        % Convergence Test
        DTC = abs ((TC_opt-TC_old)/TC_opt )* 100
        IF DTC < Tol
                % Display and save the result
                WRITE  xijt_opt, TC_opt
                SAVE xijt_opt, TC_opt
                RETURN
        ENDIF

        % Update new layout and total cost
        xijt0 = xijt_opt
        TC_old = TC_opt

END

IF DTC > Tol
        % Display warning
        WRITE warning message – iter_max reached, no converged solution
        % Display and save the result
        WRITE  xijt_opt, TC_opt
        SAVE xijt_opt, TC_opt
ENDIF
```

Figure 7 Pseudo Code of Combinatorial Optimization

2.3 Combinatorial Method: Approximate Benders' Decomposition and Approximate Dynamic Programming

The concept of combinatorial method based on Approximate Benders' decomposition and Approximate Dynamic Programming is proposed in this research. The basic principle for the combinatorial method is to use Approximate Benders' to sampling layouts for Approximate Dynamic Programming. To enhance the method further, the layouts from all period are combined into one large sample set for Approximate Dynamic Programing.

The combinatorial method can be summarized step-by-step for MATLAB as in Figure 7.

2.4 Logic-Based Model (Hooker (2000))

A logic-based formulation of the DQAP uses a variable $Y_{it} \in \{1, 2,.., n\}$ to represent the facility *i* assigned to location Y_{it}, at period *t*. The constraint that each facility is assigned exactly once to the location, and vice versa, can be written as a set of disequations:

$$Y_{it} \neq Y_{kt} \qquad , \forall i,k \text{ with } i \neq k \text{ at period } t$$

It requires that $Y_{it},...,Y_{nt}$ be a permutation of 1,2,3,…,*n* at period *t.* These constraints can be only a single global constraint for each period as

$$\text{all-different } \{Y_{1t}, Y_{2t}, Y_{3t},..., Y_{nt}\} \qquad \text{at period } t$$

where *n* is a number of facilities and locations of the problem, and *t* is a number of considering time periods.

Therefore, the logic-based model of DQAP is much more compact as:

$$\text{Minimize} \qquad \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{t=1}^{T} F_{ikt} D_{y_{it}y_{kt}} + \sum_{i=1}^{n}\sum_{t=1}^{T-1} R_{Y_{it},Y_{i(t+1)}} \tag{25}$$

$$\begin{aligned} \text{Subject to} \qquad & \text{all-different}\{Y_{11}, Y_{21}, Y_{31}, ..., Y_{n1}\} \\ & \text{all-different}\{Y_{12}, Y_{22}, Y_{32}, ..., Y_{n2}\} \\ & \vdots \\ & \text{all-different}\{Y_{1T}, Y_{2T}, Y_{3T}, ..., Y_{nT}\} \end{aligned} \tag{26}$$

where $C_{ijklt} = F_{ikt} \times D_{Y_{it},Y_{kt}}$

It is possible to demonstrate that the logic-based model of DQAP is equivalent to DQAP. Let the DQAP defined in (20) to (24) be designated problem *Q*, and the logic-based model of DQAP defined in (25) to (26) be designated problem *L*. The following theorem assures the equivalence of *Q* and *L* for any given set of cost coefficients.

Theorem 3: The feasible solutions of problems *Q* and *L* can be placed in one-to-one correspondence with equal values of the cost functions. A feasible solution $Y^{(Q)}$ of *Q* corresponds to a feasible solution $(Y^{(L)}, \textit{all-different(Y)})$ of *L* if and only if $Y^{(Q)} = Y^{(L)}$.

Proof: It is sufficient to show that the constraints of problem *L* are such that for any given permutation matrix $Y^{(L)}$ at a given period *t*, *Y* at period *t* are determined uniquely by the relations

$$\text{all-different}\{Y_{1t}, Y_{2t}, Y_{3t}, ..., Y_{nt}\} \qquad \text{at period } t$$

where $Y_{it} \in \{1, 2, .., n\}$ to represent the facility *i* assigned to location and period Y_{it}. Since the constraint for each period t requires that $Y_{1t}, Y_{2t}, Y_{3t}, ..., Y_{nt}$ all take distinct values. It covers the idea that each facility is assigned exactly once to the location and vice versa. These relations are equivalent to

$$Y_{it} \neq Y_{kt} \qquad , \forall i,k \text{ with } i \neq k \quad \text{at period } t$$

.

It follows immediately as

$$Y_{it} \neq Y_{kt} \Leftrightarrow Y_{i(y_{it})k(y_{kt})t}$$

$$Y_{i(y_{it})k(y_{kt})t} \geq X_{iy_{it}t} + X_{ky_{kt}t} - 1 \text{ that } Y_{i(y_{it})k(y_{kt})t} = Y_{ijklt} = 1 \Rightarrow X_{ijt} = X_{klt} = 1$$

$$M_{i(y_{it})(y_{i(t+1)})(t+1)} \geq X_{i(y_{it})} + X_{i(y_{i(t+1)})} - 1 \text{ that } M_{i(y_{it})(y_{i(t+1)})(t+1)} = M_{ijl(t+1)} = 1 \Rightarrow X_{ijt} = X_{il(t+1)} = 1$$

In order to prove the converse, let $X_{ijt} = X_{klt} = 1$. Then, from the constraint of ILP

$$Y_{ijklt} \geq X_{ijt} + X_{klt} - 1$$

$$M_{ijl(t+1)} \geq X_{ijt} + X_{il(t+1)} - 1$$

As proof in Theorem 1: then

$$Y_{ijklt} = 1$$

whenever $X_{ijt} = X_{klt} = 1$

Since,

$$X_{ijt} = X_{klt} = 1 \Rightarrow Y_{ijklt} = 1 = Y_{i(y_{it})k(y_{kt})t}$$

From the assignment constraints, therefore

$$Y_{i(y_{it})} \neq Y_{k(y_{kt})}$$

And the logic-based constraint can take over to

all-different $\{Y_{1t}, Y_{2t}, Y_{3t}, ..., Y_{nt}\}$ at period t Q.E.D.

In this form, there is only one constraint with ($n \times T$) real variables, and it consists of entirely checkable constraints and search variables. A logic-based method can treat the all-different constraint directly without converting it to inequalities as well as constraint satisfaction seeks a feasible solution to a set of constraints. However, checkable constraints belong to NP. Because a checkable constraint can be evaluated only after a solution is specified, values for its variable must be enumerated.

The structural algorithm of finding solution is therefore to branch on the search variables. It is impractical to keep branching until all search variables are determined. Logical inference as domain reduction algorithms can be applied to the checkable constraint before the variable domains becomes singletons.

Constraint logic programming (CLP) is a way of implementing the constraint satisfaction since integer programming methods cannot deal with an all-different constraint, $\{Y_{11},Y_{21},\ldots, Y_{n1},\ldots, Y_{nt}\}$. It uses a programming language to specify at least the outline how the problem is to be solved. In this research, the ECLiPSe is introduced in order to solve the logic-based problem. ECLiPSe, the software system, is firstly developed by IC-Parc (the Centre for Planning and Resource Control at Imperial College London) but IC-Parc has closed down in November 2005. ECLiPSe continues to be owned, used, and sponsored by Cisco Technology Inc.

A tree search method in ECLiPSe is directly the way to find the solution without adding special inference. Especially, constraint propagation enforcing arc-consistency for general array expression, developed by Brand S.(2001), is applied as the array constraint library to the program.

RESULTS AND DISCUSSION

This chapter is divided into two main parts:

1. Approximate Benders' decomposition (ABD) and Approximate Dynamic Programming (ADP) based methods and
2. Logic-based method.

ABD and ADP based methods are gradually developed from the conventional Benders' decomposition to ABD and ADP method with trust-region and successive adaptation procedure. The flowchart of Benders' decomposition algorithm is shown in appendix B. The MATLAB source code of the combinatorial method (ABD+ADP with the trust-region constraint) with successive adaptation procedure used in the study is given in appendix C. The methods are tested extensively both with small-scale problems ($n \leq 6$ and $T \leq 8$) and large-scale problems ($n = 20, 40$ and $T = 3, 5$). F_{ikt}, D_{jlt} and R_{ijlt} for the DQAP of $n \leq 40$ and $T \leq 8$ are given in appendix A. It is assume that R_{ijlt} remains unchanged over different periods. The result and discussion are presented in section 1 to 7.

Logic-based method is another competitive alternative and promising method for a large-scale DQAP problem. The ECLiPSE source code of the method in the study is given in appendix D. The method is preliminarily tested with small-scale problems in order to assess its performance. The result and discussion are presented in section 8.

1. Benders' Decomposition (BD)

DQAP of the size of $n = 3$ and $T = 2$ is solved by Benders' decomposition. An initial layout is assumed to be arbitrary and set to be $X_{iit} = 1, \forall i, t$. The result is shown in Table 1. The method needs to add 34 cuts before the solution converges (lower bound = upper bound). The number of all possible combinations of the problem is $(N!)^T = (3!)^2 = 36$. It is evident that BD has to add almost all the cuts before it reaches the

optimal layout. By checking the layout produced during each cut of Benders' decomposition, the optimal layout is actually found at cut 16 where the upper bound first reaches the optimal total cost. Clearly Benders' decomposition adds more than 100% unnecessary cuts before the solution converges. The same conclusion stands qualitatively for other small-scale problems (not shown). Benders' decomposition seems to add cuts randomly without any bias even though bias toward optimal solution is desired.

Table 1 The result for DQAP of size $n = 3$, $T = 2$ by BD (LB = UB)

Benders Cuts	Layout	Cost (units)	result
Initial layout:	123,123	23,148	Elapsed time is 22.939333 seconds.
BD is at cut: 1 & LB/UB: -23148 23148	231,312	20,943	Number of cuts = 34
BD is at cut: 2 & LB/UB: -20943 20943	312,231	22,679	Lower and upper bounds: 13377 13377
BD is at cut: 3 & LB/UB: -6887 20943	213,132	18,349	mean(z_sub)-3 sigma(z_sub): 9556.31
BD is at cut: 4 & LB/UB: -5162 18349	132,213	23,470	mean(z_sub): 20211
BD is at cut: 5 & LB/UB: -4614 18349	321,321	14,913	Sigma(z_sub): 3551.5633
BD is at cut: 6 & LB/UB: 236 14913	312,123	19,556	**The best total cost = 13,377**
BD is at cut: 7 & LB/UB: 2601 14913	213,213	20,264	**The best layout:**
BD is at cut: 8 & LB/UB: 2601 14913	213,321	16,367	**Period 1: 2 3 1**
BD is at cut: 9 & LB/UB: 3391 14913	321,132	17,157	**Period 2: 3 2 1**
BD is at cut: 10 & LB/UB: 3391 14913	321,213	19,139	
BD is at cut: 11 & LB/UB: 3428 14913	312,321	13,508	
BD is at cut: 12 & LB/UB: 3563 13508	132,321	19,311	
BD is at cut: 13 & LB/UB: 4025 13508	231,123	19,362	
BD is at cut: 14 & LB/UB: 4386 13508	312,132	15,646	
BD is at cut: 15 & LB/UB: 5374 13508	**231,321**	**13,377**	
BD is at cut: 16 & LB/UB: 5670 13377	123,312	24,796	
BD is at cut: 17 & LB/UB: 6337 13377	231,132	15,426	
BD is at cut: 18 & LB/UB: 6359 13377	312,213	17,600	
BD is at cut: 19 & LB/UB: 6804 13377	312,312	20,875	
BD is at cut: 20 & LB/UB: 7213 13377	123,231	26,533	
BD is at cut: 21 & LB/UB: 7213 13377	231,231	22,418	
BD is at cut: 22 & LB/UB: 7657 13377	132,132	21,226	
BD is at cut: 23 & LB/UB: 8420 13377	231,213	17,564	
BD is at cut: 24 & LB/UB: 8539 13377	123,132	19,342	
BD is at cut: 25 & LB/UB: 9304 13377	123,321	17,388	
BD is at cut: 26 & LB/UB: 9475 13377	321,123	21,015	
BD is at cut: 27 & LB/UB: 9788 13377	123,213	21,391	
BD is at cut: 28 & LB/UB: 10955 13377	321,312	22,440	
BD is at cut: 29 & LB/UB: 11226 13377	213,123	22,181	
BD is at cut: 30 & LB/UB: 12555 13377	132,312	26,797	
BD is at cut: 31 & LB/UB: 12589 13377	321,231	24,110	
BD is at cut: 32 & LB/UB: 12704 13377	213,312	23,695	
BD is at cut: 33 & LB/UB: 12780 13377	132,123	25,188	
BD is at cut: 34 & LB/UB: 13377 13377	231,321	13,377	

From the result, the performance of Benders decomposition has relatively poor convergence rate and is highly skeptic for large-scale problem application unless some accelerating techniques are introduced into the method.

2. Incomplete Benders' Decomposition (IBD)

From the previous section, BD has already found the optimal layout at cut 16^{th} for $n = 3$ and $T = 2$ problem. Therefore Benders' decomposition can stop after cut 16^{th} and report the optimal layout and total cost. However, the number of cuts that the optimal layout is first found in generic problem cannot be pre-determined. Hence a systematic stopping criterion has to be established.

As discussed in the previous section, Benders' decomposition seems to add cuts randomly without any bias suggesting that the layout sampling process of Benders' decomposition seems to be random. By making such assumption, the sampling process of BD is characterized by Gaussian distribution. Consequently, the total cost of layout (the cost function of the sub-problem) produced at each cut must be distributed normally. A stopping criterion may then be developed systematically based on such assumption.

According to the Gaussian distribution, if the lower bound is greater than or equal to the difference between the average of total cost and the 3 sigma of its average $\left(\hat{\mu}\text{-}3\hat{\sigma}\right)$:

$$\text{LB} \geq \text{Average of Total Cost}_{\text{sample layouts}} - 3\ \text{Standard Deviation}_{\text{sample layouts}}$$

Benders' decomposition should contain the first 0.001349% of population on the lower side of the cost and can be terminated. Other searching method can then applied to search for the optimal layout.

The result of the new stopping criterion is imposed in BD to the problem of the size of $n = 3$ and $T = 2$ is shown in Table 2. Now, BD terminates at cut 25 – only about 3/4 of the number of cuts required by an exact condition (LB =UB). The layout sample set generated by BD has already contained the optimal layout (at cut 16). The same conclusion holds qualitatively for other small-scale problems (not shown).

Table 2 The result for DQAP of size $n = 3$, $T = 2$ by IBD ($LB \geq \mu_{z\text{-sub}} - 3\sigma_{z\text{-sub}}$)

Benders Cuts	Layout	Cost (units)	Result
Initial layout:	123,123	23,148	Elapsed time is 15.118509 seconds.
BD is at cut: 1 & LB/UB: -23148 23148	231,312	20,943	Number of cuts = 25
BD is at cut: 2 & LB/UB: -20943 20943	312,231	22,679	Lower and upper bounds: 9304 13377
BD is at cut: 3 & LB/UB: -6887 20943	213,132	18,349	mean(z_sub)-3 sigma(z_sub): 9035.2289
BD is at cut: 4 & LB/UB: -5162 18349	132,213	23,470	mean(z_sub): 19318.76
BD is at cut: 5 & LB/UB: -4614 18349	321,321	14,913	sigma(z_sub): 3427.8437
BD is at cut: 6 & LB/UB: 236 14913	312,123	19,556	**The best total cost = 13,377**
BD is at cut: 7 & LB/UB: 2601 14913	213,213	20,264	**The best layout:**
BD is at cut: 8 & LB/UB: 2601 14913	213,321	16,367	**Period 1: 1 2 3,**
BD is at cut: 9 & LB/UB: 3391 14913	321,132	17,157	**Period 2: 3 2 1**
BD is at cut: 10 & LB/UB: 3391 14913	321,213	19,139	
BD is at cut: 11 & LB/UB: 3428 14913	312,321	13,508	
BD is at cut: 12 & LB/UB: 3563 13508	132,321	19,311	
BD is at cut: 13 & LB/UB: 4025 13508	231,123	19,362	
BD is at cut: 14 & LB/UB: 4386 13508	312,132	15,646	
BD is at cut: 15 & LB/UB: 5374 13508	**231,321**	**13,377**	
BD is at cut: 16 & LB/UB: 5670 13377	123,312	24,796	
BD is at cut: 17 & LB/UB: 6337 13377	231,132	15,426	
BD is at cut: 18 & LB/UB: 6359 13377	312,213	17,600	
BD is at cut: 19 & LB/UB: 6804 13377	312,312	20,875	
BD is at cut: 20 & LB/UB: 7213 13377	123,231	26,533	
BD is at cut: 21 & LB/UB: 7213 13377	231,231	22,418	
BD is at cut: 22 & LB/UB: 7657 13377	132,132	21,226	
BD is at cut: 23 & LB/UB: 8420 13377	231,213	17,564	
BD is at cut: 24 & LB/UB: 8539 13377	123,132	19,342	
BD is at cut: 25 & LB/UB: 9304 13377	123,321	17,388	

The new stopping criterion accelerates IBD considerably and may now be imposed for large-scale problems. However, a more systematic searching method is required before large-scale problems can be attempted.

3. Incomplete Benders' Decomposition (IBD) and Approximate Dynamic Programming (ABD)

Despite there is many ways to search for optimal layout from a given sample set sampled by IBD and ADP has been documented to be one of the most promising methods especially for large-scale problems. The concept of combinatorial method based on IBD and ADP is tested in this section. The basic principle for the combinatorial method is to use IBD to sampling layouts for ADP. To enhance the method further, the layouts from all period are combined into one large sample set for ADP.

The combinatorial method is tested for $n = 3$, $T = 2$ DQAP problem with an initial layout; $X_{iit} = 1$; $\forall i,t$. The result is shown in Table 3. It is clear that ADP successfully finds an optimal layout from the samples from IBD. A closer look suggests that the number of layout for ADP is 6 which are equal to all possible combinations for any given period, $n! = 3! = 6$. Indeed for this small-scale problem, ADP is an exact DP. The test on other small-scale problems (not shown) shows that IBD generates all possible combinations for any given period; ADP is an exact DP.

Test of the combinatorial method against large-scale problems should shed light on its true performance. Nevertheless the test is impossible because of the mixed-integer linear programming MATLAB function in master problem solver in IBD spends considerably long time to progress through each cut and many times fails to function correctly. Move from MATLAB to CPLEX for ILP or other programming languages may release the problem since they are specifically in the discipline of optimization. Nevertheless, the limited performance of MILP will hold the method from truly large-scale problems.

Table 3 The result for DQAP of size $n = 3$, $T = 2$ by IBD and ADP

Benders Cuts	Layout	Cost (Units)	Result
Iteration 1			
Initial layout:	123,123	23,148	**Elapsed time is 16.007747 seconds.**
BD is at cut: 1 & LB/UB: -23148 23148	231,312	20,943	**LB >= mean(z_sub) – 3 sigma (z_sub)**
BD is at cut: 2 & LB/UB: -20943 20943	312,231	22,679	Number of cuts = 25
BD is at cut: 3 & LB/UB: -6887 20943	213,132	18,349	Lower and upper bounds: 9304 13377
BD is at cut: 4 & LB/UB: -5162 18349	132,213	23,470	mean(z_sub)-3 sigma(z_sub): 9035.2289
BD is at cut: 5 & LB/UB: -4614 18349	321,321	14,913	mean(z_sub): 19318.76
BD is at cut: 6 & LB/UB: 236 14913	312,123	19,556	sigma(z_sub): 3427.8437
BD is at cut: 7 & LB/UB: 2601 14913	213,213	20,264	***** Dynamic Programming *****
BD is at cut: 8 & LB/UB: 2601 14913	213,321	16,367	Number of layout samples 6
BD is at cut: 9 & LB/UB: 3391 14913	321,132	17,157	**The best total cost = 13,377**
BD is at cut: 10 & LB/UB: 3391 14913	321,213	19,139	**The best layout:**
BD is at cut: 11 & LB/UB: 3428 14913	312,321	13,508	**2 3 1,**
BD is at cut: 12 & LB/UB: 3563 13508	132,321	19,311	**3 2 1**
BD is at cut: 13 & LB/UB: 4025 13508	231,123	19,362	***Reduction in the total cost from***
BD is at cut: 14 & LB/UB: 4386 13508	312,132	15,646	***the previous iteration (%): 73.0433***
BD is at cut: 15 & LB/UB: 5374 13508	**231,321**	**13,377**	
BD is at cut: 16 & LB/UB: 5670 13377	123,312	24,796	
BD is at cut: 17 & LB/UB: 6337 13377	231,132	15,426	
BD is at cut: 18 & LB/UB: 6359 13377	312,213	17,600	
BD is at cut: 19 & LB/UB: 6804 13377	312,312	20,875	
BD is at cut: 20 & LB/UB: 7213 13377	123,231	26,533	
BD is at cut: 21 & LB/UB: 7213 13377	231,231	22,418	
BD is at cut: 22 & LB/UB: 7657 13377	132,132	21,226	
BD is at cut: 23 & LB/UB: 8420 13377	231,213	17,564	
BD is at cut: 24 & LB/UB: 8539 13377	123,132	19,342	
BD is at cut: 25 & LB/UB: 9304 13377	123,321	17,388	

4. Approximate Benders' Decomposition (ABD) and Approximate Dynamic Programming (ADP)

An alternative taken in this work is rather aggressive. The solution of mixed integer linear programming problem in master problem of IBD is approximated by linear programming whose solution is a real number between from 0 to 1. Hungarian method is then applied to round the solution to 0 or 1 and ensures that the final layout from the approximation satisfies assignment problem constraint. The cost function defined in Hungarian method is the different between solutions from linear programming to 1 – the larger the different from 1, the more expensive cost for assigning facility to that location. Since the new solution method for master problem is based entirely on approximation, IBD should now be referred to as ABD.

The approximation is so aggressive that there is no guarantee that ABD will find optimal layout after adding all the cuts. Therefore a pure searching for optimal layout from samples from ABD may not work. However, optimal layout (or at least close-to optimal layout) from ABD may be determined after ADP has been performed on samples from ABD sampling process. Iterative procedure for ABD and ADP may be introduced to enhance the solution method. One main advantage of ABD is the computational time required to solve the problem reduces by at least a factor of ten. Clearly the combinatorial method – ABD and ADP – is truly ideal for large-scale DQAP problem.

The test for $n = 3$ and $T = 2$ DQAP (Table 4) is carried out. ABD adds only 19 cuts before the lower bound is greater than or equal to the z-value of total cost of –3. ADP starts and it finds the optimal layout. Notice that ADP takes 6 random layouts generated from ABD for every period. Obviously the optimal solution has to be found because there are only $3! = 6$ possible combinations for each period. ABD for small-scale problems seems to generate all possible combinations for each period; No conclusion can be drawn from these small-scale problems

The proposed method is also tested for $n = 5$ and 6 with $T = 1$, 5 and 8. The result is shown in Table 5. Different trials result from different parameter setting i.e. the number of maximum allowable cuts and/or initial solutions. The method could find the optimal solution for every problem. A closer look to the number of layout for ADP suggests that ABD samples all possible layout combinations for the problem of $n = 5$ and more than 60% of all possible layout combinations for the problem of $n = 6$. Clearly ADP should have no difficulty identifying an optimal solution for each problem. In addition, it is interesting to note that ABD terminates with different conditions for different number of maximum allowable cuts. Nevertheless, ABD generates enough layout samples covering more than 60% of all possible layout combination. The optimal solution could be easily obtained.

Table 4 The result for DQAP of size $n = 3$, $T = 2$ by ABD and ADP

Benders Cuts	Layout	Cost (Units)	Result
Iteration 1			
Initial layout:	123,123	23,148	**Elapsed time is 16.007747 seconds.**
BD is at cut: 1 & LB/UB: -23148 23148	231,312	20,943	**LB >= mean(z_sub) – 3 sigma (z_sub)**
BD is at cut: 2 & LB/UB: -20943 20943	312,231	22,679	Number of cuts = 25
BD is at cut: 3 & LB/UB: -6887 20943	213,132	18,349	Lower and upper bounds: 9304 13377
BD is at cut: 4 & LB/UB: -5162 18349	132,213	23,470	mean(z_sub)-3 sigma(z_sub): 9035.2289
BD is at cut: 5 & LB/UB: -4614 18349	321,321	14,913	mean(z_sub): 19318.76
BD is at cut: 6 & LB/UB: 236 14913	312,123	19,556	sigma(z_sub): 3427.8437
BD is at cut: 7 & LB/UB: 2601 14913	213,213	20,264	*** **Dynamic Programming** ***
BD is at cut: 8 & LB/UB: 2601 14913	213,321	16,367	Number of layout samples 6
BD is at cut: 9 & LB/UB: 3391 14913	321,132	17,157	**The best total cost = 13,377**
BD is at cut: 10 & LB/UB: 3391 14913	321,213	19,139	**The best layout:**
BD is at cut: 11 & LB/UB: 3428 14913	312,321	13,508	**2 3 1,**
BD is at cut: 12 & LB/UB: 3563 13508	132,321	19,311	**3 2 1**
BD is at cut: 13 & LB/UB: 4025 13508	231,123	19,362	***Reduction in the total cost from***
BD is at cut: 14 & LB/UB: 4386 13508	312,132	15,646	***the previous iteration (%): 73.0433***
BD is at cut: 15 & LB/UB: 5374 13508	**231,321**	**13,377**	
BD is at cut: 16 & LB/UB: 5670 13377	123,312	24,796	
BD is at cut: 17 & LB/UB: 6337 13377	231,132	15,426	
BD is at cut: 18 & LB/UB: 6359 13377	312,213	17,600	
BD is at cut: 19 & LB/UB: 6804 13377	312,312	20,875	
BD is at cut: 20 & LB/UB: 7213 13377	123,231	26,533	
BD is at cut: 21 & LB/UB: 7213 13377	231,231	22,418	
BD is at cut: 22 & LB/UB: 7657 13377	132,132	21,226	
BD is at cut: 23 & LB/UB: 8420 13377	231,213	17,564	
BD is at cut: 24 & LB/UB: 8539 13377	123,132	19,342	
BD is at cut: 25 & LB/UB: 9304 13377	123,321	17,388	
Iteration 2			LB >= mean(z_sub) - 3 sigma (z_sub)
BD is at cut: 1 & LB/UB: -13377 13377			Number of cuts = 19
BD is at cut: 2 & LB/UB: -13377 13377			Lower and upper bounds: 6926 13377
BD is at cut: 3 & LB/UB: -5493 13377			mean(z_sub)-3 sigma(z_sub): 6853.9937
BD is at cut: 4 & LB/UB: -5357 13377			mean(z_sub): 20403.3158
BD is at cut: 5 & LB/UB: -2540 13377			sigma(z_sub): 4516.4407
BD is at cut: 6 & LB/UB: -2188 13377			*** Dynamic Programming ***
BD is at cut: 7 & LB/UB: -141 13377			Number of layout samples 6
BD is at cut: 8 & LB/UB: 1777 13377			**The best total cost = 13,377**
BD is at cut: 9 & LB/UB: 2188 13377	*not show the layouts and cost*		**The best layout:**
BD is at cut: 10 & LB/UB: 2540 13377			**2 3 1,**
BD is at cut: 11 & LB/UB: 2540 13377			**3 2 1**
BD is at cut: 12 & LB/UB: 2799 13377			***Reduction in the total cost from***
BD is at cut: 13 & LB/UB: 3648 13377			***the previous iteration (%):* 0**
BD is at cut: 14 & LB/UB: 4056 13377			
BD is at cut: 15 & LB/UB: 5695 13377			**Optimisation has converged at iteration 2**
BD is at cut: 16 & LB/UB: 5728 13377			**Optimal total cost = 13377**
BD is at cut: 17 & LB/UB: 5728 13377			**Optimal layout =**
BD is at cut: 18 & LB/UB: 6257 13377			**2 3 1,**
BD is at cut: 19 & LB/UB: 6926 13377			**3 2 1.**

Table 5 The result for DQAP of size n = 5, 6 and T = 1, 5, 8 by ABD and ADP

Size	Ncutmax	#Cuts MP	Iter	Int. L/O	Terminated by	#L/O DP	Best Cost (Units)	found opt. sol.	ExeTime (sec.)	%Reduce
N5, T1	**100**	27	1	opt	LB ≥ μ-3σ	27	41864	✓	1.749036	0
N5,T5	**100**	100	1	opt	Ncutmax	120	174210	✓	27.803327	0
	200	200	1	opt	Ncutmax	120	174210	✓	81.056714	0
	300	219	1	opt	LB ≥ μ-3σ	120	174210	✓	94.186254	0
	300	170	1	(I)	LB ≥ μ-3σ	120	174210		162.22784	24.9435
		219	2		LB ≥ μ-3σ	120	174210	✓		0
N5,T8	**100**	100	1	opt	Ncutmax	120	278342	✓	60.066174	0
	200	200	1	opt	Ncutmax	120	278342	✓	182.840525	0
	300	300	1	opt	Ncutmax	120	278342	✓	365.709657	0
	400	364	1	opt	LB ≥ μ-3σ	120	278342	✓	525.250426	0
	400	389	1	(I)	LB ≥ μ-3σ	120	278342		1093.43988	26.0683
		364	2		LB ≥ μ-3σ	120	278342	✓		0
N6, T1	**100**	43	1	Opt	LB ≥ μ-3σ	43	66645	✓	2.903234	0
N6,T5	**50**	50	1	(I)	max_nCuts	212	289012		45.565614	19.6943
		50	2		max_nCuts	359	286636			0.82893
		50	3		max_nCuts	483	288636	✓		0
	100	100	1	Opt	max_nCuts	374	283651	✓	47.654302	0
	200	200	1	Opt	max_nCuts	546	283651	✓	148.485748	0
	300	300	1	Opt	max_nCuts	622	283651	✓	296.60055	0
	400	400	1	Opt	max_nCuts	650	283651	✓	514.537402	0
	500	403	1	Opt	LB ≥ μ-3σ	650	283651	✓	505.4232	0
N6,T8	**42**	42	1	(I)	max_nCuts	269	462296		84.45462	19.9279
		42	2		max_nCuts	449	456368			1.299
		42	3		max_nCuts	549	456368	✓		0
	48	48	1	(I)	max_nCuts	269	462296		106.785208	19.9279
		48	2		max_nCuts	486	456368			1.299
		48	3		max_nCuts	577	456368	✓		0
	1000	1000	1	opt.	max_nCuts	719	456368	✓	7361.04358	0

5. Approximate Benders' Decomposition (ABD) and Approximate Dynamics Programming (ADP) – Large-Scale Problem Testing

To obtain a better picture of the performance of the proposed combinatorial method on large-scale problem, the problem with n = 20, T = 5 and n = 40, T = 3 are tested. Two parameters for the proposed combinatorial method to be tested are:

- the number of maximum allowable cuts in Approximate Benders' decomposition (section 5.1), and
- the initial layout (section 5.2).

5.1 The Number of Maximum Allowable Cuts in Approximate Benders' Decomposition

With this large-scale problem, ABD has a great difficulty pushing the lower bound to be greater than or equal to the z-value of total cost of –3 after several hundreds of cuts. Therefore the number of maximum allowable cuts in Approximate Bender decomposition is limited to 10 to 500 to make the method more practical for large-scale problems. With large-scale problems, Approximate Benders' decomposition is likely to terminate when the number of cuts reaches the maximum allowable cuts; the quality of the sample layouts for ADP may degraded considerably. Iterative procedure is introduced to the method to further improve the total cost of the solution. The iteration continues until

$$\frac{\text{Total Cost}_{\text{current iteration}} - \text{Total Cost}_{\text{previous iteration}}}{\text{Total Cost}_{\text{previous iteration}}} \times 100 \le \text{tolerance}$$

Preliminary numerical experiment suggests that the solution is insensitive to the tolerance for iterative procedure; set to 0.001% to all tests.

The first set of test shown in Table 6 is performed on DQAP problem of the size $n = 20$ and $T = 5$ over the number of maximum allowable cuts of 5, 10, 20, 40 and 500. Note that these numbers of cuts are negligibly small compared to the total number of all possible combinations $(20!)^5$. The initial layout is set to be an arbitrary; $X_{iit} = 1, \forall i, t$. The general trend is that the total cost of layout solution improves as the number of maximum allowable cuts increases except when the number of maximum allowable cuts is set to 40. Notice that the number of sample layouts for ADP also increases successively as the number of maximum allowable cuts increases. The reason that solution for the number of maximum allowable cuts of 40 does not improve compared to that of 20 is not clear but it seems to closely relate to a small number of iterations. Further reduction of the tolerance for the iteration fails to add more iterations; no improvement in the total cost.

The best layout solution obtained when the number of maximum allowable cuts is set to 500. The method takes 366,872.16 seconds (see Table 6). The total cost of that layout is 4,416,613. The solution from the proposed combinatorial method compares well with other metaheuristic methods i.e. simulated annealing, genetic algorithm and tabu search (see Table 7) (Muenvanichakul 1998). The total cost of best layout solution from the proposed method is less than 1% higher than the total cost from simulated annealing and tabu search with random initial layout. The proposed method produces better solution than genetic algorithm with random initial layout by about 5%. However, tabu search and genetic algorithm quickly outperform the proposed method if the best layout solution from simulated annealing is supplied in as initial layouts for the two methods. Tabu search generates the best solution whose total cost is about 4% lower than that of the proposed method while genetic algorithm generates solution whose total cost is about 1% lower than that of the proposed method. The proposed method may perform equally well if a better layout solution is supplied as an initial layout. The possibility of taking advantage of a better initial layout in the proposed method is investigated further.

The performance of the method is quickly checked against the DQAP problem of the size $n = 40$ and $T = 3$. The number of maximum allowable cuts in the test is set to 500. The calculation terminates during iteration 4 where the solution has not converged yet. The termination is a result of "out-of-memory" error during ADP routine appending a new set of layouts from ABD to existing layouts in ADP (4501 layouts). Better memory management and layout selection may be implemented in the future to improve the performance of the method. Nevertheless, the result from iteration 3 (as shown in Table 7) indicates that indeed the proposed combinatorial method is truly comparable to other metaheuristic methods. The best solution from the proposed method with a given arbitrary layout is 11,203,203 which is slightly less than that from tabu search and genetic algorithm with given random initial layouts. Nevertheless, simulated annealing still performs better in terms of the total cost than the proposed method by less than 5%. By taking advantage of initial layout from simulated annealing, the proposed method may not be able to produce a better total

cost compared to tabu search and genetic algorithm. The investigation is carried out in the next section.

Table 6 The DQAP of size $n = 20$, $T = 5$ by ABD and ADP

Ncutmax	ExeTime (sec.)	Iter	#Cuts MP	Terminated by	#L/O DP	Best Cost	found better L/O	%Reduce
5	18.57	1	5	max_nCuts	26	4,568,677		0.74
		2	5	max_nCuts	51	4,550,667		0.40
		3	5	max_nCuts	74	4,550,667	×	0.00
10	118.74	1	10	max_nCuts	51	4,546,671		0.26
		2	10	max_nCuts	101	4,541,365		0.12
		3	10	max_nCuts	150	4,533,156		0.18
		4	10	max_nCuts	200	4,529,629		0.08
		5	10	max_nCuts	250	4,525,737		0.09
		6	10	max_nCuts	299	4,519,097		0.15
		7	10	max_nCuts	337	4,519,097	×	0.00
20	171.02	1	20	max_nCuts	101	4,545,772		0.24
		2	20	max_nCuts	201	4,494,312		1.15
		3	20	max_nCuts	301	4,488,038		0.14
		4	20	max_nCuts	400	4,485,903		0.05
		5	20	max_nCuts	463	4,485,903	×	0.00
40	640.11	1	40	max_nCuts	201	4,534,442		0.01
		2	40	max_nCuts	401	4,494,882		0.88
		3	40	max_nCuts	601	4,494,882	×	0.00
500	366872.16	1	500	max_nCuts	2501	4,472,102		4.98
		2	500	max_nCuts	5001	4,434,723		0.84
		3	500	max_nCuts	7501	4,429,065		0.13
		4	500	max_nCuts	9981	4,428,254		0.02
		5	500	max_nCuts	12459	4,416,613		0.03
		6	500	max_nCuts	14954	4,416,613	×	0.00
500*	47552.52	1	500	max_nCuts	2505	4,256,480	✓	0.00

***** Initial layout: SA

Table 7 Cost comparisons of large-scale DQAP problems among four algorithms

Problem Size	Cost (units) from Simulated Annealing (SA)	Cost (units) from Tabu Search Method		Cost (units) from Genetic (w/o RW)		Cost (units) from ABD+ADP
		Initial L/O: Random	Initial L/O: from SA	Initial L/O: Random	Initial L/O: from SA	
N=20,T=5	4,383,877	4,399,513	4,256,480	4,605,719	4,361,559	4,416,613
N=40,T=3	10,872,483	11,245,315	10,761,358	11,358,612	10,854,512	11,203,203

5.2 Initial Layout

The initial layout for the proposed combinatorial method is set to the best layout obtained from simulated annealing from Luangpaiboon (1995). The numbers of maximum allowable cuts are set to be 10, 20, 40 and 500 to test in the DQAP problem of the size $n = 20$ and $T = 5$. The result shown in Table 8 indicates that the proposed method fails to improve the solution from simulated annealing, in contrast to genetic algorithm or tabu search (Muenvanichakul 1998). The method stops after the first iteration suggesting that the method indeed does not take advantage of a better initial layout. The root of the failure of the proposed method is tracked down to poor characteristics of Benders' decomposition – a cutting plane algorithm. It is well-known that cutting plane algorithms usually requires an impractically large number cuts to determine the solution because in the early iterations, the solutions tend to oscillate wildly from on region of the feasible set to another; hence slow down the convergence rate of the algorithm (Hiriart-Urruty and Lemaréchal, 1996). The result from the test with 500 maximum allowable cuts for Approximate Benders' decomposition shown in Table 8 clearly confirms the statement. The mean value of the total cost of the sampled layout (the total cost from sub-problem) is 4,738,904.87 whereas the total cost of the initial layout (best solution from simulated annealing) is only 4,383,729 which is 8.80 times the standard deviation below the mean value. Clearly, Approximate Benders' decomposition samples layout randomly through out the feasible set without any bias toward the initial layout in particular when the number of maximum allowable cuts is much smaller than the number of all possible combinations; taking no advantage of a good initial layout. A quick test to the problem of the size $n = 40$ and $T = 3$ (as shown in Table 9) also confirms the conclusion. The proposed method terminates after the solution converges at the first iteration; no solution improvement.

Table 8 The result for DQAP of size n = 20, T = 5 by ABD+ADP (Initial Solution from SA)

		ABD					ADP		%	Exe. Time
Ncutmax	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (units)	Reduction	(sec.)
10	1	-4324372	4383729	4353107	4715185.3	120692.7	55	4,383,729	0	7.68
20	1	-4229535	4383729	4456340	4728735.5	90798.6	105	4,383,729	0	15.10
40	1	-4044747	4383729	4523621	4727538.1	67972.5	205	4,383,729	0	137.93
80	1	-3913040	4383729	4560880	4730936.5	56685.4	405	4,383,729	0	526.51
500	1	-3432137	4383729	4617816	4738904.9	40362.9	2505	4,383,729	0	39,845.42

Table 9 The result for DQAP of size n = 40, T = 3 by ABD+ADP (Initial Solution from SA)

		ABD					ADP		%	Exe. Time
Ncutmax	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (units)	Reduction	(sec.)
10	1	-10653081	10871848	10820554.5	11497181.2	225542.2	33	10,871,848	0	27.39
20	1	-10653081	10871848	11035033.9	11528162.3	164376.1	63	10,871,848	0	54.89
40	1	-10653081	10871848	11185279.4	11548592.5	121104.4	123	10,871,848	0	114.40
80	1	-10304811	10871848	11267233.6	11552749.4	95171.9	243	10,871,848	0	360.43
500	1	-9804727	10871848	11365189.5	11560153.9	64988.1	1503	10,871,848	0	6773.06

In order to improve the proposed combinatorial method further, Trust-region method or any other accelerating methods to Benders' decomposition (Santoso et al 2004) may be implemented to the proposed combinatorial method. A proper control of sampling process should result in the method that takes a full advantage of a good initial layout. In this work, the implementation of trust-region to regulate the sampling process in the master problem is explored.

6. Approximate Benders' Decomposition and Approximate Dynamics Programming with Trust-Region Constraint

An additional constraint for trust-region to regulate the change of the solutions in the two consecutive iterations in the master problem is introduced into the master problem of the Approximate Benders' decomposition in the proposed combinatorial method. As proposed by Santoso *et al* (2004), the additional constraint imposed at p+1th iteration bounds the Hamming distance of the master problem solution at p+1th iteration from the solution at p^{th} iteration:

$$\sum_{q \in X^p} \left(1 - x_q^{p+1}\right) + \sum_{q \notin X^p} x_q^{p+1} \leq \Delta^{p+1},$$

where Δ^{p+1} is the trust-region size at p+1$^{\text{th}}$ iteration, X^p is the master problem solution obtained at p^{th} iteration and let $X^p = \{q : x_q^p = X_{ijt}^p = 1\}$. For early iterations, the trust-region size should be small to minimize the oscillation of the solution. As the number of cuts in the master problem constraint increases, the trust-region size should increase in order to relax the trust-region constraint. However, in this work, the trust-region size is kept constant through out each solution procedure. The choice of constant trust-region is reasonable in this work. The number of maximum allowable cuts is several orders of magnitude smaller than the total of all possible cuts; the solution is likely to wildly oscillate without any trust-region constraint. Adaptive trust-region size is subjected to future study and not further pursued in this work. The example is shown in Figure 8.

The proposed combinatorial method with trust-region constraint is tested with n = 20, T = 5 problem. The maximum number of allowable cuts is set to 20. The trust-region size is set to be 0.5 corresponding to allowing only 25% of the layout to be changed. The best layout from simulated annealing (Luangpaiboon 1995) is used as an initial layout. The result shown in Table 10 indicates that the trust-region constraint improves the solution and has a great potential to combine with other methods such as simulated annealing to improve the solution. The total cost from the proposed method with the initial layout from simulated annealing is 4,360,788. It is slightly better than that of genetic algorithm but approximately 2% higher than that of tabu search. Nevertheless, the proposed method with trust-region constraint only takes 712.42 seconds.

Example: N=3, T=1

Assume layout changes totally at iteration p+1 e.g.

Iteration p

$i \backslash j$	1	2	3
1	**1**		
2			**1**
3		**1**	

Iteration p+1

$i \backslash j$	1	2	3
1		**1**	
2	**1**		
3			**1**

$$X^{p} = \{1,3,2 : X_{11}^{p} = X_{23}^{p} = X_{32}^{p} = 1\}$$

$$X^{p+1} = \{2,1,3 : X_{121}^{p+1} = X_{211}^{p+1} = X_{331}^{p+1} = 1\}$$

Then, the TR constant is

$$\sum_{q \in X^{p}} (1 - x_q) + \sum_{q \notin X^{p}} x_q$$

$$= \{(1 - X_{111}^{p+1}) + (1 - X_{231}^{p+1}) + (1 - X_{321}^{p+1})\} + \{(X_{121}^{p+1} + X_{131}^{p+1}) + (X_{211}^{p+1} + X_{221}^{p+1}) + (X_{311}^{p+1} + X_{331}^{p+1})\}$$

$$= \{(1-0) + (1-0) + (1-0)\} + \{(1+0) + (1+0) + (1+0)\}$$

$$= 6$$

$$= 2(\mathrm{N})$$

$$= \Delta^{p+1} \quad \text{(TR-Size)}$$

The TR-Size is 6, it means that the constraint will allow all layouts to be changed. By restricting this bound only some layouts can be changed depending on the size of trust region.

Figure 8 How the trust region constraint works

The same test with trust-region size of 0.5 and number of maximum allowable cuts of 20 is done with the problem of the size n = 40 and T = 3. The result shown in Table 11 confirms that the trust-region constraint indeed improves the solution. The total cost from the proposed method is 10,834,881 well comparable to genetic algorithm and tabu search – slightly better than that of genetic algorithm but approximately 1% higher than that of tabu search. The method takes only 361.09 seconds to complete the calculation.

It is clear from the result that trust-region constraint improves the performance of the proposed method in particular from the point of view of taking advantage of a good initial layout. It is worthwhile to explore the possibility of implementing the

method successively with different trust-region size and number of maximum allowable cuts.

Table 10 The result for DQAP of size n = 20, T = 5 by ABD+ADP (TR= 0.5 and Ncutmax =20)

		ABD					ADP		%	Exe. Time
Ncutmax	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost	Reduction	(sec.)
20	1	1951171	4383729	4386527	4489488	34320.5	74	4,379,134	0.105	
	2	1893463	4379134	4375619	4470737	31705.8	134	4,363,194	0.365	
	3	1898569	4363194	4363632	4470117	35495.1	196	4,362,946	0.006	
	4	1868946	4362946	4371028	4481903	36958.2	257	4,360,788	0.049	
	5	1850283	4360788	4382254	4481025	32923.7	317	4,360,788	0	712.426

Table 11 The result for DQAP of size n = 40, T = 3 by ABD+ADP (TR= 0.5 and Ncutmax =20)

		ABD					ADP		%	Exe. Time
Ncutmax	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost	Reduction	(sec.)
40	1	5015544	10862694	10903468	11109815	68782.29	58	10,848,451	0.1313	
	2	4891148	10848451	10879847	11080321	66824.69	118	10,834,881	0.1252	
	3	5019301	10834881	10878983	11071842	64286.28	178	10,834,881	0	892.3669

7. Successive Adaptation Procedure

To refine the method further, the proposed method, Approximate Benders' decomposition and Approximate Dynamic Programming with trust-region constraint, is implemented successively with different parameters. For a given initial layout, the procedure starts with a small number of maximum allowable cuts and large trust-region size. The solution is then supplied as an initial layout for the method with a smaller trust-region size to regulate the sampling process in ABD to be in the neighbor of a good initial layout from the previous step. The procedure continues successively until no further improvement is achievable or termination by the user. The procedure may continue successively with a larger number of maximum allowable cuts with a successive trust-region size reduction procedure. Pseudo code shown in Figure 9 shows the procedure implemented in this study. The procedure may be further fine tuned to improve its performance.

```
% Successive Adaptation Procedure
% Combinatorial Optimization – Benders' decomposition + Dynamic
%       Programming for Dynamic Quadratic Assignment Problem

% Read initial layout
READ xijt0                                  % initial layout

% Start the procedure
% Loop over the number of maximum allowable cuts (increasing)
FOR Ncut_max = 10, 20, 40, 80

        % Loop over the trust-region size (decreasing)
        FOR TR_size_constant = 1, 0.5, 0.25

                % Call Combinatorial Optimization Method – ABD + ADP
                CALL BENDERS_DP         % Input: xijt0( initial layout)
                                        % Output:xijt_opt( best layout), TC_opt

                % Update layout
                xijt0 = xijt_opt

        END
END
```

Figure 9 Pseudo code for successive adaptation procedure

Table 12 and Figure 10 show the result of the proposed method with the successive adaptation procedure starting from the best layout from simulated annealing with $n = 20$ and $T = 5$. The total cost from the procedure is 4,316,387 and the grand total cost reduction over the whole procedure is 1.5362%. The total cost from the procedure compares well with genetic algorithm and tabu search (see Table 14) (Muenvanichakul 1998) starting from the same initial layout. The total cost of best layout solution is less than genetic algorithm by about 1% while it is higher than tabu search by about 1%. The CPU time of the whole procedure is 11,459.10 seconds which is longer than genetic algorithm and tabu search but is still comparable to both methods. Fine tuning the procedure certainly improves the procedure. As it can be seen from Figure 10 many steps do not improve the solution. The most two effective steps produce the cost reduction of 0.542 and 0.469% respectively occurring at the number of maximum allowable cuts of 40 and trust-region size of 0.25 and 0.5 respectively. The most two effect steps indeed account for about 2/3 of the grand total cost reduction. Unlike the first few steps running at the number of maximum allowable cuts of 10 and 20, they seem not to improve the solution so much. It seems that with good initial layout, the procedure requires sufficient number of maximum allowable cuts before it can produce significant improvement in the total cost when

the trust-region size is relatively restricted. The figure also suggests that there is not much improvement at the number of maximum allowable cuts of 80. It is speculated that with this even better solution, the procedure requires more number of maximum allowable cuts than 80. Skipping these unnecessary steps certainly cut down CPU time and improves its speed performance.

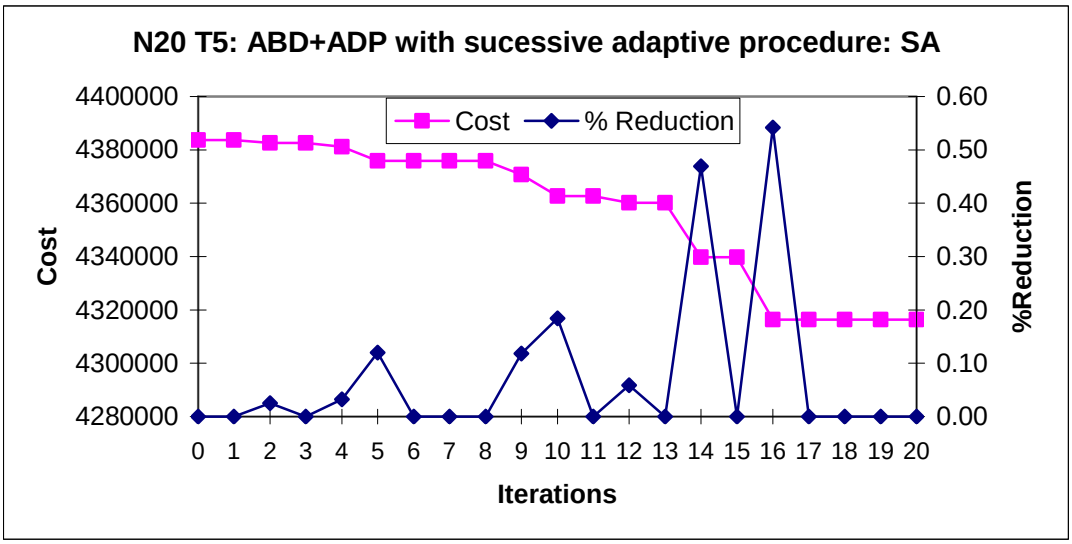


Figure 10 The result for $n = 20$, $T = 5$ by ABD+ADP with Successive Adaptation Procedure (Initial L/O: SA)

Table 12 The result for DQAP of size $n = 20$, $T = 5$ by ABD+ADP with the successive adaptation procedure

		Initial L/O	ABD						ADP		%	
Ncutmax	TRsize	previous iter	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (units)	Reduction	Result
10	1	4383729	1	-564227	4383729	4353209	4592088.9	79626.8	46	4,383,729	0	Exe.Time (sec.)
	0.5	4383729	1	1820051	4383729	4360546	4479421.2	39624.9	42	4,382,619	0.025	**11,459.10074**
			2	1834930	4382619	4351774	4488289.2	45505.1	72	4,382,619	0	**Optimal total cost = 4,316,387**
	0.25	4382619	1	3153378	4382619	4353698	4425577.0	23959.5	38	4,381,192	0.033	Optimal layout =
			2	3094024	4381192	4360007	4428497.2	22829.9	63	4,375,929	0.120	[2 10 4 14 15 8 12 19 5 9 18
			3	3177974	4375929	4348676	4427297.0	26206.9	79	4,375,929	0	7 16 20 1 13 11 3 17 6,
20	1	4375929	1	-572114	4375929	4431009	4627969.7	65653.7	86	4,375,929	0	
	0.5	4375929	1	1837776	4375929	4383596	4490024.3	35476.2	70	4,375,929	0	15 19 5 6 7 16 9 8 17 13 20
	0.25	4375929	1	3222037	4375929	4367004	4431194.6	21397.0	64	4,370,756	0.118	11 14 3 1 10 18 2 12 4,
			2	3176792	4370756	4362813	4439094.6	25427.3	104	4,362,711	0.184	
			3	3179003	4362711	4355411	4435499.5	26696.2	143	4,362,711	0	16 2 10 20 6 5 18 7 8 13 14
40	1	4362711	1	-568966	4362711	4465412	4630351.9	54980.0	166	4,360,144	0.059	1 3 19 15 4 17 11 9 12,
			2	-559676	4360144	4458310	4618990.7	53560.3	330	4,360,144	0	
	0.5	4360144	1	1885594	4360144	4383580	4470473.4	28964.6	134	4,339,784	0.469	14 11 17 16 7 10 4 8 19 2
			2	1944711	4339784	4380123	4469063.3	29646.8	254	4,339,784	0	15 5 9 12 3 20 18 13 1 6,
	0.25	4339784	1	3232382	4339784	4325389	4396194.9	23602.0	106	4,316,387	0.542	
			2	3227543	4316387	4328220	4393795.7	21858.6	194	4,316,387	0	13 14 5 2 3 20 10 6 9 12 1
80	1	4316387	1	-428257	4316387	4484380	4633614.3	49744.7	327	4,316,387	0	11 16 19 17 4 18 8 7 15]
	0.5	4316387	1	2039470	4316387	4393004	4482505.2	29833.9	257	4,316,387	0	**Grand Total Improvement (%):**
	0.25	4316387	1	3272865	4316387	4335230	4399481.5	21417.2	210	4,316,387	0	**1.5362**

Table 13 The result for DQAP of size $n = 40$, $T = 3$ by ABD+ADP with the successive adaptation procedure

		Initial L/O	ABD						ADP		%	
Ncutmax	TRsize	previous iter	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost	Reduction	Result
10	1	10862694	1	-845855	10862694	10797359	11331452.8	178031.4	33	10,862,694	0	Exe.Time (sec.)
	0.5	10862694	1	4782656	10862694	10818836	11096055.5	92406.5	33	10,848,451	0.1313	**16,359.3187**
			2	4722455	10848451	10809278	11065574.8	85432.1	63	10,834,881	0.1252	
			3	5019301	10834881	10813157	11066558.9	84467.3	93	10,834,881	0	**Optimal total cost =10,817,002**
	0.25	10834881	1	8117750	10834881	10825647	10931496.5	35283.2	29	10,834,881	0	
20	1	10834881	1	-869250	10834881	10954706	11353399.7	132898.0	63	10,834,881	0	
	0.5	10834881	1	5019301	10834881	10878983	11071841.5	64286.3	63	10,834,881	0	
	0.25	10834881	1	8117750	10834881	10824530	10920072.7	31847.4	49	10,831,734	0.0291	
			2	8102980	10831734	10822888	10917369.1	31493.8	95	10,824,821	0.0639	
			3	8061377	10824821	10785505	10912518.2	42337.7	136	10,817,002	0.0723	
			4	7930768	10817002	10816894	10912327.8	31811.1	175	10,817,002	0	
40	1	10817002	1	-872283	10817002	11070731	11440652.8	123307.1	123	10,817,002	0	
	0.5	10817002	1	5142014	10817002	10885564	11057379.0	57271.7	123	10,817,002	0	
	0.25	10817002	1	8085454	10817002	10829653	10920002.0	30116.2	103	10,817,002	0	**Grand Total of Improvement**
80	1	10817002	1	-838800	10817002	11165485	11444728.3	93081.1	243	10,817,002	0	**(%):**
	0.5	10817002	1	5510650	10817002	10903194	11051771.9	49526.0	243	10,817,002	0	**0.4206**
	0.25	10817002	1	8332945	10817002	10840570	10923444.4	27624.8	181	10,817,002	0	

The proposed combinatorial method with trust-region constraint and the same successive adaptation procedure is also tested with n = 40 and T = 3 problem as shown in Table 13 and Figure 11. The total cost from the procedure is 10,817,002 and the grand total cost reduction over the whole procedure is 0.4206%. The total cost from the procedure compares well with genetic algorithm and tabu search (see Table 14) (Muenvanichakul 1998) starting from the same initial layout. The total cost is approximately 0.4% lower than that from genetic algorithm and slightly higher than that from tabu search by only 0.5%. The method spends 16,359.33 seconds to arrive at the solution which is comparable to other methods. Speed up can be gained by optimizing the procedure as can be seen from Figure 10. However at this problem size the procedure tends to favor a small number of maximum allowable cuts of 10 and 20 with restricted trust-region size of 0.25 and 0.5. The reduction of these steps accounts for almost 100% of the grand total cost reduction. Again, there is no clear pattern or system to cut out unnecessary steps.

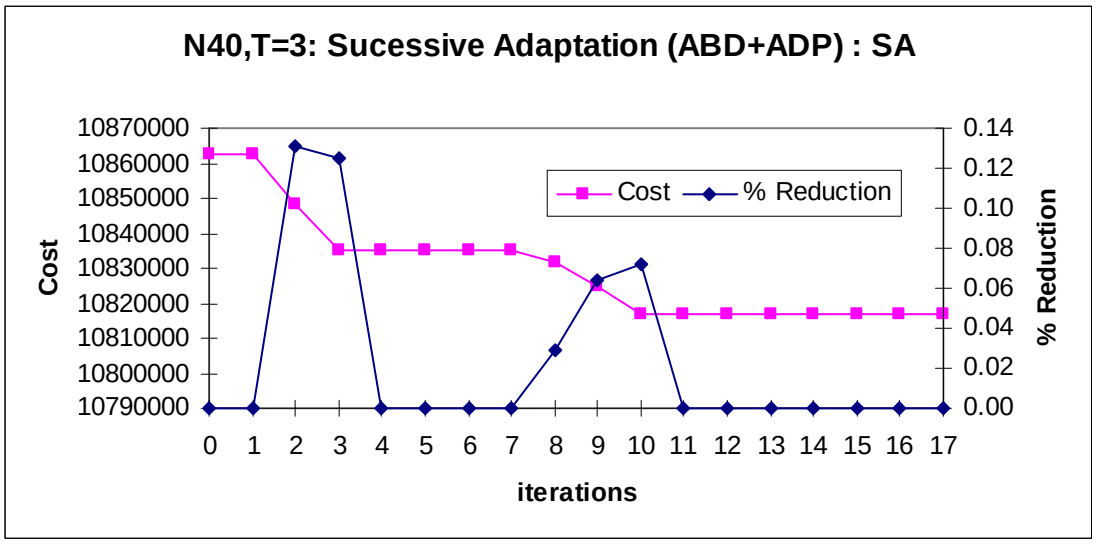


Figure 11 The result for n = 40, T = 3 by ABD+ADP with Successive Adaptation Procedure (Initial L/O: SA)

Detail results are shown in Tables 15 to 19 and Figures 12 to 16. In contrast to the problem starting from best layout simulated annealing as discussed in the previous section, the method evolves in a more order fashion. A total cost is greatly reduced for low number of maximum allowable cuts for a large trust-region and its reduction

decrease as trust-region decreases. The total cost reduction greatly reduces again when the number of maximum allowable cuts increases even though the total cost reduction is not as large as the previous number of maximum allowable cuts. This trend continues as the procedure goes. It is clear that initially with random layout the method greatly takes advantage of ADP to single out the best solution from random layout generated by ABD with a large trust-region. With the reduction of trust-region further, ABD tends to sample layout only in the neighbor of a best layout from the previous step. As a result the solution from ADP is only slightly improved from the previous step solution. If the true global minimum lays further way from this neighbor, the proposed method and procedure may not be able to single out the true global minimum.

Table 14 Cost comparisons of large-scale DQAP problems among four different algorithms

		Tabu Search Method		**Genetic (w/o RW)**		**ABD+ADP (Sucessive)**		**ABD+ADP**
Problem Size	**Simulated Annealing**	**Initial L/O: Random**	**Initial L/O: from SA**	**Initial L/O: Random**	**Initial L/O: from SA**	**Initial L/O: arbitrary L/O**	**Initial L/O: from SA**	
N=20,T=5	4,383,877	4,399,513	4,256,480	4,605,719	4,361,559	4,370,302	4,316,387	4,416,613
N=40,T=3	10,872,483	11,245,315	10,761,358	11,358,612	10,854,512	11,015,498	10,817,002	11,203,203

Table 15 The result for DQAP of size $n = 20$, $T = 3$ by ABD+ADP with the successive adaptation procedure

		Cost of Initial (Previous)	ABD						ADP			
Ncutmax	TRsize	Layout	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (Units)	% Reduction	Result
10	1	2,904,999	1	-335166	2828438	2802411	2892693.2	30094.0759	31	2,817,583	3.1025	Exe.Time (sec.)
			2	-400225	2817583	2786425.9	2877236.3	30270.1317	61	2,807,924	0.3440	**8,335.322407**
			3	-356978	2807924	2776214.2	2891248.5	38344.7832	121	2,803,015	0	
	0.5	2,803,015	1	1187836	2780440	2766349.4	2819496.6	17715.7440	31	2,770,797	1.1628	**Optimal total cost = 2,724,991**
			2	1176806	2770797	2747166.2	2803349.3	18727.6925	58	2,759,928	0.3938	Optimal layout =
			3	1188104	2759928	2747611.3	2812801.1	21729.9220	82	2,754,314	0.2038	
			4	1174798	2754314	2739056.7	2810927.8	23957.0187	107	2,754,314	0	[13 10 3 4 18 12 7 1 9 15
	0.25	2,754,314	1	2042369	2754314	2733499.9	2771005.2	12501.7693	23	2,741,067	0.4833	11 5 2 14 6 19 17 8 16 20,
			2	2025456	2741067	2729471.9	2773427.7	14651.9471	31	2,741,067	0	
20	1	2,741,067	1	-332001	2741067	2755119.1	2875953.2	40278.0319	63	2,741,067	0	14 2 6 10 9 17 7 8 3 11 12
	0.5	2,741,067	1	1271160	2741067	2730713.6	2801577.0	23621.1496	62	2,739,138	0.0704	16 13 4 15 5 1 18 19 20,
			2	1311132	2739138	2726890.5	2804556.8	25888.7646	114	2,735,901	0.1183	
			3	1215516	2735901	2732109.4	2806483.8	24791.4781	157	2,735,901	0	12 15 17 18 5 2 7 11 16 9
	0.25	2,735,901	1	2178963	2735901	2724734	2764693.0	13319.6791	44	2,731,638	0.15606	13 14 3 8 20 1 6 4 19 10]
			2	2060212	2731638	2726550.2	2762354.9	11934.8933	59	2,731,638	0	
40	1	2,731,638	1	-189971	2731638	2777165.8	2871661.3	31498.4919	123	2,731,638	0	
	0.5	2,731,638	1	1333157	2731638	2734904.1	2803175.7	22757.1999	117	2,730,292	0.0493	
			2	1317209	2730292	2729860	2796819.9	22319.9536	220	2,725,034	0.1930	
			3	1346484	2725034	2735301.3	2796097.9	20265.5237	312	2,725,034	0	**Grand Total Improvement (%):**
	0.25	2,725,034	1	2151709	2725034	2722173.4	2758758.3	12194.9691	69	2,724,991	0.0016	**6.1965**
			2	2261445	2724991	2721102.1	2759975.7	12957.8576	88	2,724,991	0	
80	1	2,724,991	1	-178092	2724991	2782996.9	2870554.1	29185.7224	243	2,724,991	0	
	0.5	2,724,991	1	1378867	2724991	2724991	2799260.5	17701.1235	226	2,724,991	0	
	0.25	2,724,991	1	2261445	2724991	2723147	2762257.2	13036.7353	96	2,724,991	0	

Table 16 The result for DQAP of size $n = 20$, $T = 5$ by ABD+ADP with the successive adaptation procedure

		Cost of Initial (Previous)	ABD						ADP			
Ncutmax	TRsize	Layout	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (units)	% Reduction	Result
10	1	4,694,840	1	-597887	4599495	4584722	4681535.2	32271.0884	48	4,545,184	3.293	Exe.Time (sec.)
			2	-611566	4545184	4507059	4641238.4	44726.3151	94	4,519,383	0.571	**18,238.799**
			3	-648196	4519383	4484716	4648703.4	54662.6159	136	4,511,835	0.167	
			4	-636835	4511835	4470717	4649927.6	59736.9230	176	4,510,612	0.027	**Optimal total cost = 4,370,302**
			5	-640884	4510612	4490543	4645806.7	51754.6351	216	4,481,067	0.659	Optimal layout =
			6	-625306	4481067	4458107	4640266.1	60719.5774	256	4,473,579	0.167	[9 15 19 5 3 10 20 11 16
			7	-595410	4473579	4447564	4645770.6	66068.8414	296	4,473,579	0	14 8 6 18 7 2 4 17 12 13 1,
	0.5	4,473,579	1	1825161	4473579	4453841	4518659.0	21606.1183	46	4,447,118	0.595	
			2	1856113	4447118	4423506	4523807.3	33433.8552	82	4,437,603	0.214	18 4 11 14 19 13 12 15 5 16
			3	1836073	4437603	4391264	4515683.5	41473.3077	119	4,434,884	0.061	20 9 3 10 1 17 6 8 7 2,
			4	1807073	4434884	4387462	4502034.5	38190.9166	154	4,422,816	0.273	
			5	1793770	4422816	4409667	4511803.3	34045.5725	188	4,422,816	0	19 14 16 5 2 3 6 1 13 15
	0.25	4,422,816	1	3142731	4420090	4389856	4450141.7	20095.3494	33	4,402,198	0.468	18 11 8 7 4 20 17 10 12 9,
			2	3070938	4402198	4376210	4444774.1	22854.5639	52	4,402,037	0.004	
			3	3255021	4402037	4377962	4442724.5	21587.5124	70	4,395,918	0.139	11 18 3 1 4 9 17 15 20 10
			4	3188118	4395918	4372412	4437537.7	21708.4938	85	4,391,543	0.100	14 16 2 13 12 7 8 5 19 6,
			5	3165699	4391543	4369426	4437369.3	22647.8557	103	4,387,784	0.086	
			6	3160296	4387784	4359618	4438934.0	26438.8327	122	4,387,784	0	20 15 11 9 3 13 18 5 17 12
20	1	4,387,784	1	-574715	4387784	4434129	4651549.3	72473.3518	90	4,387,784	0	4 7 19 1 10 8 14 2 16 6]
	0.5	4,387,784	1	1969994	4387784	4397916	4514724.0	38936.1401	88	4,387,784	0	
	0.25	4,387,784	1	3192892	4387784	4376957	4440045.9	21029.5194	60	4,380,287	0.17	**Grand Total Improvement (%):**
			2	3172362	4380287	4377990	4442004.8	21338.2860	102	4,380,287	0	**6.9127**
40	1	4,380,287	1	-613123	4380287	4485132	4633296.0	49388.0318	170	4,380,287	0	
	0.5	4,380,287	1	1966131	4380287	4421555	4510757.9	29734.1513	164	4,380,249	0.001	
			2	2158335	4380249	4422765	4519396.2	32210.3145	323	4,380,249	0	
	0.25	4,380,249	1	3284461	4380249	4387958	4446184.7	19408.9067	105	4,374,209	0.138	
			2	3286929	4374209	4382314	4448102.9	21929.6523	200	4,371,260	0.067	
			3	3267990	4371260	4376133	4448465.1	24110.6186	274	4,370,302	0.022	
			4	3282404	4370302	4385036	4448088.7	21017.6523	316	4,370,302	0	
80	1	4,370,302	1	-507597	4370302	4496348	4640322.7	47991.6787	330	4,370,302	0	
	0.5	4,370,302	1	2096097	4370302	4423003	4513024.6	30007.2628	309	4,370,302	0	
	0.25	4,370,302	1	3366067	4370302	4391351	4453223.0	20624.0332	188	4,370,302	0	

Table 17 The result for DQAP of size $n = 20$, $T = 8$ by ABD+ADP with the successive adaptation procedure

		Cost of Initial (Previous)	ABD						ADP			
Ncutmax	TRsize	L/O	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (units)	% Reduction	Result
10	1	7,491,424	1	-990774	7371529	7306020.4	7447508.4	71510.29	76	7,226,091	3.6719	Exe.Time (sec.)
			2	-1065424	7226091	7176632	7391162.9	71510.29	136	7,221,690	0.0609	**122,621.4244**
			3	-1106983	7221690	7183239.8	7396882.5	71214.24	195	7,213,114	0.1189	
			4	-1037375	7213114	7176030.3	7393299.6	72423.10	255	7,157,562	0.7761	**Optimal total cost = 6,887,656**
			5	-982937	7157562	7112735.8	7385020.8	90761.68	316	7,157,562	0	Optimal layout =
	0.5	7,157,562	1	2826148	7157562	7121551.3	7248807.8	42418.83	60	7,114,094	0.6110	[16 18 13 4 15 8 7 2 9 19
			2	2785234	7114094	7069482.6	7250731.7	60416.36	115	7,096,603	0.2465	10 1 11 14 6 3 17 5 12 20,
			3	2739016	7096603	7062363.9	7270537.5	69391.19	169	7,096,603	0	
	0.25	7,096,603	1	5017123	7090843	7035053.2	7145859.1	36935.29	50	7,040,483	0.7971	17 12 11 16 18 9 3 19 20 6
			2	5016529	7040483	7018526.1	7151375.0	44282.95	84	7,040,483	0	15 13 8 10 2 14 4 7 1 5,
20	1	7,040,483	1	-975930	7040483	7130613.5	7426367.3	98584.59	129	7,040,483	0	
	0.5	7,040,483	1	2826923	7040483	7092535.7	7264116.1	57193.45	122	7,035,718	0.0677	18 14 3 19 10 16 7 1 15 9
			2	2835275	7035718	7077332.4	7262398.7	61688.74	226	7,035,718	0	12 11 17 13 8 4 2 20 6 5,
	0.25	7,035,718	1	5072550	7035718	7049934.1	7153658.9	34574.93	92	7,035,718	0	
40	1	7,035,718	1	-965037	7035718	7186243.6	7387041.0	66932.46	245	7,035,718	0	10 12 3 2 4 15 5 6 20 19 17
	0.5	7,035,718	1	2956437	7035718	7118747.3	7262246.4	47833.02	235	7,035,718	0	9 16 18 11 14 7 8 13 1,
	0.25	7,035,718	1	5178913	7035718	7064718.9	7151856.1	29045.73	143	7,030,984	0.0673	
			2	5029469	7030984	7062710.2	7148510.5	28600.11	231	7,027,912	0.0437	18 1 3 19 20 12 7 8 17 14
			3	5152644	7027912	7062333.4	7152365.4	30010.66	295	7,019,962	0.1133	5 13 4 15 9 11 2 10 6 16,
			4	5179482	7019962	7051410.4	7153824.8	34138.14	344	7,019,962	0	
80	1	7,019,962	1	-959048	7019962	7224230.5	7404550.5	60106.66	493	6,998,411	0.3079	17 12 11 16 18 9 3 19 20 6
			2	-966664	6998411	7211895.8	7397440.0	61848.06	981	6,971,408	0.3873	15 13 8 10 2 14 4 7 1 5,
			3	-982883	6971408	7197490.1	7389298.7	63936.20	1462	6,971,408	0	
	0.5	6,971,408	1	3056482	6971408	7073080.5	7204607.0	43842.15	462	6,971,292	0.0017	18 14 3 19 10 16 7 1 15 9
			2	3005947	6971292	7080561	7197962.0	39133.66	916	6,958,694	0.1810	12 11 17 13 8 4 2 20 6 5,
			3	3129613	6958694	7031956.2	7164625.4	44223.07	1333	6,949,793	0.1281	
			4	2979491	6949793	7038920.6	7162230.1	41103.16	1765	6,916,619	0.4796	10 12 3 2 4 15 5 6 20 19
			5	2958020	6916619	7018961.8	7156809.7	45949.29	2197	6,908,915	0.1115	17 9 16 18 11 14 7 8 13 1]
			6	3048580	6908915	7043595.6	7170346.3	42250.24	2619	6,908,439	0.0069	
			7	3089027	6908439	7029524.8	7149722.3	40065.82	3049	6,908,439	0	**Grand Total Improvement (%):**
	0.25	6,908,439	1	5034370	6908439	6964927.4	7053099.1	29390.55	260	6,906,112	0.0337	**8.0595**
			2	4994794	6906112	6943191.6	7030882.8	29230.41	448	6,887,656	0.2680	
			3	5139462	6887656	6936476.1	7032453.4	31992.42	576	6,887,656	0	

Table 18 The result for DQAP of size $n = 40$, $T = 3$ by ABD+ADP with the successive adaptation procedure

		Cost of Initial (Previous)	ABD							ADP		
Ncutmax	**TRsize**	**L/O**	**Iter**	**LB**	**UB**	**μ-3σ**	**μ**	**σ**	**# L/O**	**Cost (units)**	**% Reduction**	**Result**
10	1	11,601,354	1	-937886	11447438	11363984	11506978.5	47664.8102	31	11,341,161	2.2942	Exe.Time (sec.)
			2	-961097	11341161	11223438	11472996	83185.981	61	11,307,214	0.3002	**43,910.1129**
			3	-1060163	11307214	11282061	11488000	68646.4223	91	11,264,538	0.3789	
			4	-992058	11264538	11199108	11400984.4	67292.0126	121	11,221,599	0.3827	**Optimal total cost = 11,015,498**
			5	-1067083	11221599	11159406	11445450.3	95348.2606	151	11,221,599	0	
	0.5	11,221,599	1	4815228	11221599	11158629	11307941.8	49770.9	33	11,202,722	0.1685	
			2	4843841	11202722	11150448	11291943.7	47165.1415	63	11,168,248	0.3087	
			3	4759652	11168248	11081158	11261319.6	60053.9701	93	11,119,053	0.4424	
			4	4734302	11119053	11080425	11251620.5	57065.3178	123	11,119,053	0	
	0.25	11,119,053	1	8007227	11117102	11058942	11148529.8	29862.4818	32	11,093,173	0.2333	
			2	8018103	11093173	11065638	11144959.9	26440.5217	55	11,093,173	0	
20	1	11,093,173	1	-1016822	11093173	11157851	11433344.4	91831.1002	63	11,093,173	0	
	0.5		1	5026942	11093173	11081094	11248335.05	55746.8804	63	11,090,073	0.0280	
			2	5140574	11090073	11087209	11241315.4	51368.8346	123	11,088,130	0.0175	
			3	4996158	11088130	11098985	11241194.3	47403.2642	183	11,076,908	0.1013	
			4	4991040	11076908	11060400	11239814.85	59804.8116	243	11,076,908	0	
	0.25	11,076,908	1	8218867	11076908	11048439	11118740.75	23433.8091	58	11,042,980	0.3072	
			2	8226426	11042980	11027134	11121145.5	31337.3206	107	11,031,516	0.1039	
			3	8200320	11031516	11031136	11117264.95	28709.6767	159	11,023,833	0.0697	
			4	8221047	11023833	11019786	11124068.2	34760.807	206	11,021,290	0.0231	
			5	8190327	11021290	11020311	11115737.4	31808.8659	258	11,017,879	0.0310	
			6	8007073	11017879	11015346	11127679.6	37444.3898	311	11,015,498	0.0216	
			7	7998729	11015498	11014328	11117896.5	34522.7705	358	11,015,498	0	
40	1	11,015,498	1	-934241	11015498	11169877	11427452.28	85858.5398	123	11,015,498	0	**Grand Total of Improvement (%):**
	0.5	11,015,498	1	5175960	11015498	11058425	11232734.43	58103.3063	123	11,015,498	0	**5.0499**
	0.25	11,015,498	1	8266207	11015498	11043220	11126941.08	27906.9792	113	11,015,498	0	
80	1	11,015,498	1	-934241	11015498	11229383	11424400.28	65005.8215	243	11,015,498	0	
	0.5	11,015,498	1	5845004	11015498	11092383	11232431.44	46682.7802	243	11,015,498	0	
	0.25	11,015,498	1	8476475	11015498	11052268	11137750.44	28494.0414	203	11,015,498	0	

Table 19 The result for DQAP of size $n = 40$, $T = 5$ by ABD+ADP with the successive adaptation procedure

		Cost of Initial	ABD						ADP	%	Exe. Time
Ncutmax	TRsize	(Previous) L/O	Iter	LB	UB	μ-3σ	μ	σ	# L/O	Cost (units)	Reduction	(sec.)
10	1	19,247,187	1	-1574026	18968002	18839135	19124966.4	95277.0	51	18,726,821	2.7787	Exe.Time (sec.)
			2	-1590026	18726821	18661029	18939523.4	92831.3	101	18,645,843	0.4343	**103,756.6565**
			3	-1777612	18645843	18448420	18923043.6	158208.0	151	18,595,224	0.2722	
			4	-1762677	18595224	18546538	18904322.5	119261.4	201	18,595,224	0	**Optimal total cost =18,131,694**
	0.5	18,595,224	1	8026030	18590415	18488072	18668752.5	60226.9	47	18,496,379	0.5344	
			2	7863462	18496379	18427343	18645902.3	72853.2	88	18,446,801	0.2688	[32 26 25 31 5 34 1 39 7 33
			3	7934710	18446801	18398636	18615502.3	72288.8	128	18,437,566	0.0501	23 16 40 17 38 28 10 37 6 20
			4	7962032	18437566	18326336	18607939.4	93867.8	174	18,400,266	0.2027	2 22 12 24 21 29 19 30 15 35
			5	8024769	18400266	18361378	18560101.2	66241.0	217	18,358,586	0.2270	14 27 9 11 36 3 18 4 13 8,
			6	7881296	18358586	18293071	18533024.5	79984.4	266	18,341,344	0.0940	40 2 33 13 5 6 28 8 35 10
			7	7989264	18341344	18291680	18528703.9	79008.0	316	18,312,158	0.1594	14 31 12 3 15 4 23 27 29 20
			8	7912881	18312158	18274058	18512615.5	79519.1	365	18,309,177	0.0163	21 22 34 24 36 32 19 30 37 38
			9	7863289	18309177	18230160	18539879.6	103239.8	414	18,307,261	0.0105	26 16 11 1 7 17 18 9 39 25,
			10	7971904	18307261	18275525	18502914.4	75796.5	460	18,305,130	0.0116	39 8 36 37 19 38 23 26 40 25
			11	7866498	18305130	18256508	18525596.6	89414.5	501	18,279,038	0.1427	34 15 32 28 2 14 17 31 33 16
			12	8130347	18279038	18248180	18516423.9	89414.5	544	18,279,038	0	18 10 24 6 3 5 27 11 29 13
	0.25	18,279,038	1	13069298	18279038	18224925	18363295.0	46123.3	45	18,228,020	0.2799	30 7 1 9 35 21 20 4 12 22,
			2	13164236	18228020	18191867	18342557.9	50230.2	81	18,222,497	0.0303	39 40 36 1 21 38 23 37 30 25
			3	13006449	18222497	18197157	18320952.8	41265.2	120	18,194,772	0.1524	34 14 32 3 19 28 17 26 24 16
			4	13176373	18194772	18156375	18331465.5	58363.5	157	18,184,942	0.0541	33 10 7 11 18 13 27 9 29 22
			5	13086889	18184942	18144900	18344901.3	66667.2	190	18,184,942	0	15 6 5 12 35 2 20 4 8 31,
20	1	18,184,942	1	-1626123	18184942	18388077	18953149.5	188357.4	105	18,184,942	0	17 40 33 4 14 32 11 37 34 10
	0.5	18,184,942	1	7905941	18184942	18233998	18528239.1	98080.3	97	18,184,942	0	24 30 9 16 15 26 13 18 19 2
	0.25	18,184,942	1	13226733	18184942	18200901	18349144.0	49414.5	85	18,184,942	0	21 22 8 1 35 20 27 28 29 31
40	1	18,184,942	1	-1626123	18184942	18509720	18927617.3	139299.2	205	18,184,942	0	7 39 23 6 12 36 25 38 3 5]
	0.5	18,184,942	1	8455418	18184942	18293918	18521752.8	75944.8	183	18,184,942	0	
	0.25	18,184,942	1	13226990	18184942	18227934	18351170.8	41079.0	157	18,184,942	0	
80	1	18,184,942	1	-1626123	18184942	18569235	18902457.1	111074.0	405	18,184,942	0	
	0.5	18,184,942	1	8455418	18184942	18324768	18516279.0	63836.9	361	18,184,942	0	
	0.25	18,184,942	1	13815659	18184942	18227998	18343361.5	38454.7	314	18,165,466	0.1072	**Grand Tota ofl Improvement (%):**
			2	13806660	18165466	18192938	18318501.5	41854.4	617	18,150,446	0.0828	**5.7956**
			3	13407725	18150446	18217152	18325877.0	36241.8	915	18,150,018	0.0024	
			4	13588750	18150018	18212352	18332034.1	39893.9	1184	18,131,694	0.1011	
			5	13608469	18131694	18212835	18322891.7	36685.6	1453	18,131,694	0	

Table 20 The result summary provided by ABD+DAP with the successive adaptation procedure

Problem Size n	t	Initial Layout	Cost of Initial Layout (units)	Cost of Best Cost by ABD+ADP w/ successive (units)	Average Improvement (%)	Exe Time (sec)
20	3	Arbitrary L/O	2,904,999	2,724,991	6.1965	8,335.32
20	5		4,694,840	4,370,302	6.9127	18,238.80
20	8		7,491,424	6,887,656	8.0595	122,621.42
40	3		11,601,354	11,015,498	5.0499	43,910.11
40	5		19,247,187	18,131,694	5.7956	103,756.66
20	5	Best L/O by SA	4,383,729	4,316,387	1.5362	11,459.10
40	3		10,862,694	10,817,002	0.4206	16,359.32

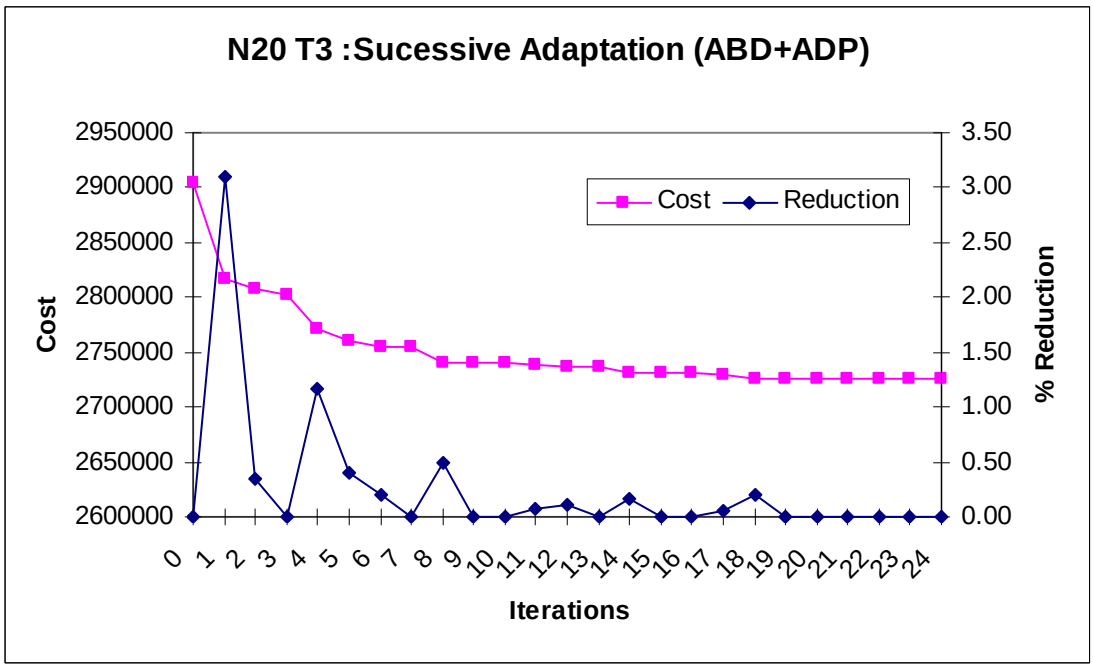


Figure 12 The result for $n = 20$, $T = 3$ by ABD+ADP with Successive Adaptation Procedure

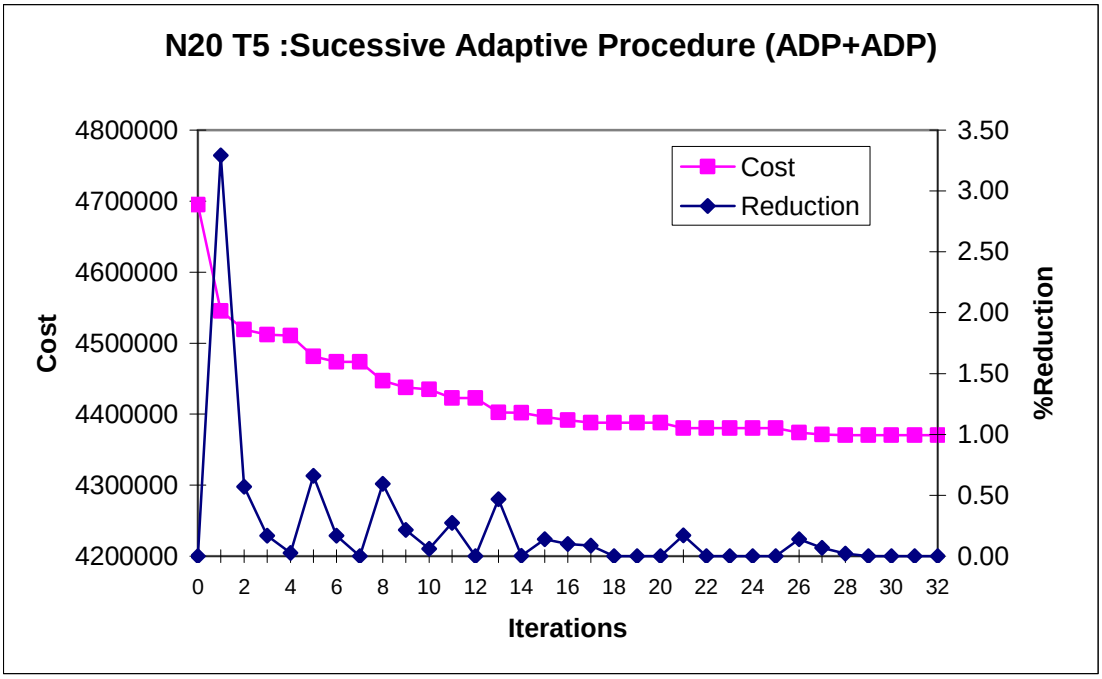


Figure 13 The result for $n = 20$, $T = 5$ by ABD+ADP with Successive Adaptation Procedure

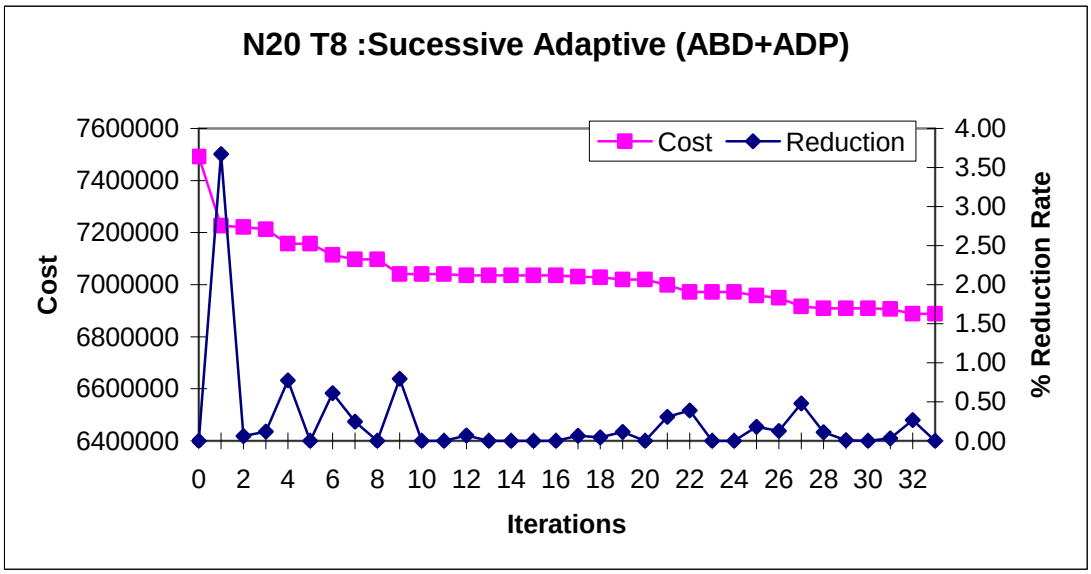


Figure 14 The result for $n = 20$, $T = 8$ by ABD+ADP with Successive Adaptation Procedure

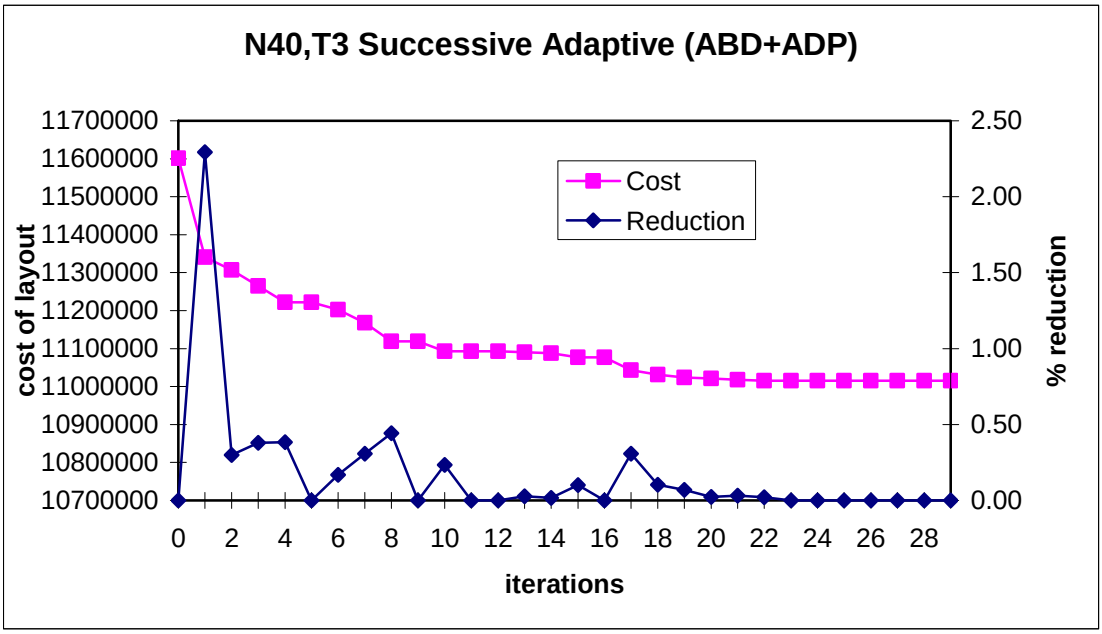


Figure 15 The result for $n = 40$, $T = 3$ by ABD+ADP with Successive Adaptation Procedure

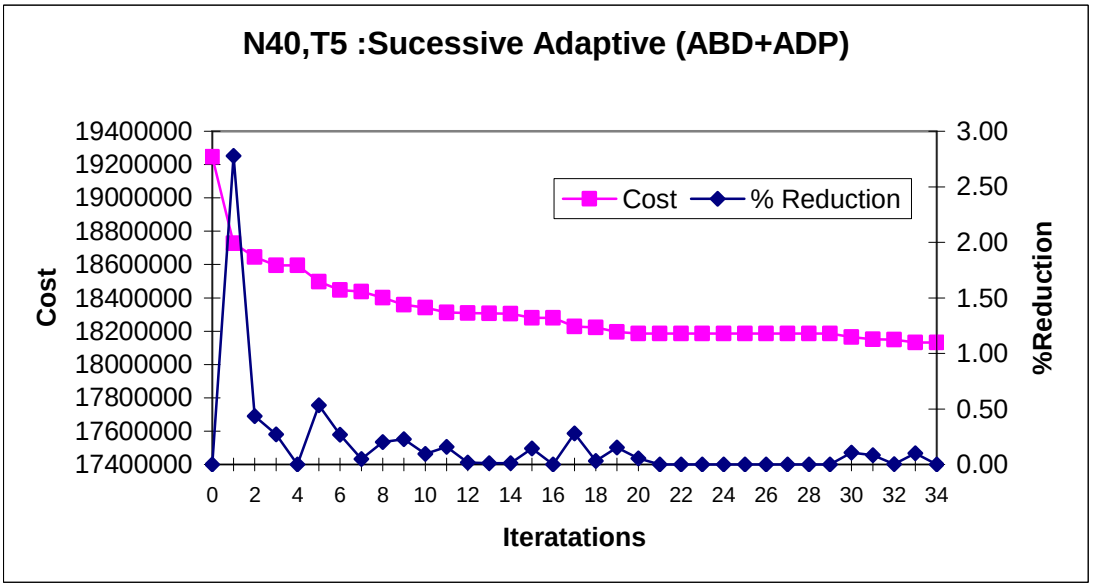


Figure 16 The result for $n = 40$, $T = 5$ by ABD+ADP with Successive Adaptation Procedure

8 Logic-Based Model

The method development of logic-based model focuses on only small-scale problems in particular for n = 3, 4, 5, 6 and T = 1, 2, 3, 4, 5, 6, 7, 8 problem to explore the optimal solutions. ECLiPSe is introduced for searching solutions of the problems and its source code in the study is given in appendix D. All problems are tested on a Pentium M 1400 MHz and 512 MB of Ram, which a windows XP-OS. Computational search for solutions must vary according to how large of problems determined by $(n!)^T$ because the constraint simply says that $Y_{11}, Y_{21}, \ldots, Y_{n1}, \ldots, Y_{nt}$ as shown in Table 21. Executing time and optimal solutions of experiments are presented in Table 22 and 23, respectively. ECLiPSe cannot search well enough when the search space is up to 2.07E+08 of problem n = 5, T = 4. After taking 39600s cpu, ECLiPSe still cannot find an optimal solution for the problem. The well performance of ECLiPSe appeared only for such very small size of problems such as n = 4, T =1, 2, 3, 4, or n = 5, T =1, 2, 3 or n = 6, T =1, 2.

Table 21 : Number of searching space

n* \ *T	**1**	**2**	**3**	**4**	**5**	**6**
4	24	576	13,824	331,776	7,962,624	1.91E+08
5	120	14,400	1,728,000	2.07E+08	2.49E+10	2.99E+12
6	720	518,400	3.73E+08	-	-	1.39E+17

Table 22 Computational time (unit: sec CPU)

n* \ *T	**1**	**2**	**3**	**4**	**5**
4	0.06	0.69	12.52	260.36	20,023
5	0.53	16.35	4,225.17	39,600*	17,968*
6	2.42	1,042.61	-	-	35,988*

* Interrupted.

Table 23 Optimal Solutions (unit: cost units)

n* \ *T	**1**	**2**	**3**	**4**	**5**
4	20,578	36,661	54,113	67,713	86,649
5	41,864	78,598	116,835	149,392*	195,785*
6	66,645	131,158	-	-	320,698*

* Interrupted

There are two types of problems as small and large problem. Small problems are the problems which are size n = 5, 6 and T = 5, 8. For small problem, CLP can not find the better solutions than ever found by others algorithms even dynamic programming (Luangpaiboon, 1995), simulated annealing, tabu search or genetic algorithm (Muenvanichakul, 2000). Except for the problem size n = 5, T = 8, CLP can get the better solution than GA but it took 64,800s CPU for searching time. (See Table 24 and 25)

Table 24: The computational times comparing among methods (unit: seconds CPU)

	Computational Methods			
Size of Problem	**SA**	**TS**	**GA**	**Logic-Based**
n = 5, T= 5	1,808	364	2,713	17,968*
n = 5, T = 8	5,481	964	4,173	17,997*
n = 6, T = 5	1,594	805	3,572	35,988*
n = 6, T = 8	5,726	1,956	5,413	35,993*

* Interrupted.

Table 25: The solutions (cost) comparing among methods (unit: cost units)

	Computational Methods			
Size of Problem	**SA**	**TS**	**GA**	**Logic-Based**
n = 5, T= 5	185,354	180,782	188,889	186,958
n = 5, T = 8	304,316	286,492	313,815	323,632
n = 6, T = 5	303,254	286,730	312,572	310,224
n = 6, T = 8	496,313	465,084	520,410	524,078

It has been found that searching for solutions of ECLiPSe is doing well with such very small problems. When a number of all possible permutations are getting bigger, ECLiPSe has taken more computing time but come up with less quality improving-solutions. For large problems such as problem size n = 20, T = 3, the permutations of period 2 and 3, after 900s CPU, are still the same as at the starting point (See Table 6). Only candidate solutions of period 1 were determined, and it also seemed that only its first ten facilities were in a process of determining. Moreover, Figure 1 and 2 shown that ECLiPSe found six better solutions for problem size n = 5, T = 4 within 10-hours running times since the searching spaces is getting big as 2.07E+08 permutations.

Table 26 The optimal solutions for small-size DQAP problems solving by the ECLiPSe

N	T		Exe.Time (sec.)	Cost (Units)	Layout
3	1		0.00	7,538	(2, 3, 1)
	2		0.05	13,377	(2, 3, 1), (3, 2, 1))
	3		0.14	21,296	(2, 3, 1), (3, 2, 1), (3, 2, 1)
	4		0.58	35,236	(2, 3, 1), (3, 2, 1), (3, 2, 1), (2, 3, 1)
	5		6.94	44,764	(2, 3, 1), (3, 2, 1), (3, 2, 1), (2, 1, 3), (2, 1, 3)
	6		11.88	50,722	(2, 3, 1), (3, 2, 1), (3, 2, 1), (2, 1, 3), (2, 1, 3), (3, 2, 1)
	7		47.41	58,641	(2, 3, 1), (3, 2, 1), (3, 2, 1), (2, 1, 3), (2, 1, 3), (3, 2, 1), (3, 2, 1)
	8		300.36	72,581	(2, 3, 1), (3, 2, 1), (3, 2, 1), (2, 1, 3), (2, 1, 3), (3, 2, 1), (3, 2, 1), (2, 3, 1)
4	1		0.06	20,578	(4, 3, 1, 2)
	2		0.69	36,661	(4, 3, 1, 2), (3, 2, 4, 1)
	3		12.52	54,113	(4, 3, 1, 2), (3, 2, 4, 1), (4, 1, 3, 2)
	4		260.36	67,713	(4, 3, 1, 2), (3, 2, 4, 1), (4, 1, 3, 2), (3, 2, 4, 1)
	5		20023.38	86,649	(4, 3, 1, 2), (3, 2, 4, 1), (4, 1, 3, 2), (3, 2, 4, 1), (1, 3, 2, 4)
	6		167438.14	102,755	(4, 3, 1, 2), (3, 2, 4, 1), (4, 1, 3, 2), (3, 2, 4, 1), (1, 3, 2, 4), (3, 2, 4, 1)
5	1		0.53	41,864	(5, 2, 1, 3, 4)
	2		16.03	78,598	(5, 2, 1, 3, 4), (4, 5, 3, 2, 1)
	5	*	17968.94	186,958	(5, 2, 1, 3, 4), (4, 5, 3, 2, 1), (2, 4, 5, 1, 3), (3, 5, 4, 2, 1), (5, 4, 3, 2, 1)
	8	*	17997.84	323,632	(5, 2, 1, 3, 4), (4, 5, 3, 2, 1), (2, 4, 5, 1, 3), (3, 5, 4, 2, 1), (5, 4, 3, 2, 1), (5, 4, 3, 2, 1), (5, 4, 3, 2, 1), (5, 4, 3, 2, 1)
6	1		2.21	66,645	(3, 6, 1, 2, 5, 4)
	2		1030.59	131,158	(3, 6, 1, 2, 5, 4), (3, 2, 4, 5, 6, 1)
	5	*	35988.39	310,224	(5, 2, 6, 3, 4, 1), (3, 6, 4, 5, 2, 1), (4, 5, 6, 3, 2, 1), (6, 5, 4, 3, 2, 1), (6, 5, 4, 3, 2, 1)
	8	*	35993.33	524,078	(5, 2, 6, 3, 4, 1), (3, 6, 4, 5, 2, 1), (4, 5, 6, 3, 2, 1), (6, 5, 4, 3, 2, 1), (6, 5, 4, 3, 2, 1), (6, 5, 4, 3, 2, 1), (6, 5, 4, 3, 2, 1)

CONCLUSION AND RECOMMENDATION

Conclusion

New combinatorial solution methods for large-scale dynamic quadratic assignment (DQAP) problem are developed in this study. The main bottleneck to such problem is that a linearized system of dynamic quadratic assignment problem is a relatively large system of mixed-integer linear programming problem making the problem a NP complete problem.

The principle algorithms in the new proposed combinatorial solution methods are Benders' decomposition successfully applied to large-scale mixed-integer linear programming problem and Approximate Dynamic Programming successfully applied to dynamic assignment problem. Due to slow convergence of Benders' decomposition and limited ability of a mixed-integer linear programming algorithm, Benders' decomposition is modified for large-scale DQAP problem. The modification includes Incomplete Benders' decomposition (IBD) utilizing a new stopping criterion (the lower bound is greater than or equal to the z-value of total cost of –3) & limit number of maximum allowable cuts for master problem (to 500 in maximum), and Approximate Benders' decomposition (ABD) approximating a mixed-integer linear programming by a rounding up of the solution from linear programming using Hungarian method. During the method development phase, DQAP of the size $n = 3$ and $T = 2$ is used as a sample model. The problems with $n = 5$ & 6 and $T = 1$, 5 & 8 are further tested.

For large-scale problem ($n = 20$ and $T = 5$ and $n = 40$ and $T = 3$), the method performs reasonably well compared to other metaheuristic methods i.e. simulated annealing, genetic algorithm and tabu search even though it only performs better than genetic algorithm. The study of using a good initial layout from other metaheuristic methods to accelerate and enhance the proposed combinatorial method fails to produce a promising result. The root of the failure comes from the nature of wild

oscillating solutions in early iterations of Benders' decomposition. The concept of trust-region is introduced to accelerate and enhance the proposed combinatorial method is explored. A constant trust-region size is implemented in the ABD as a master problem constraint. Trust-region accelerates and enhances the method significantly. The implementation of the successive parameter adaptation procedure (successive trust-region size reduction and number of maximum allowable cut increase) to the proposed combinatorial method further improves its performance. With a given arbitrary matrix layout; $X_{iit} = 1, \forall i, t$, the method is then further tested on other large-scale problems including n = 20, 40 and T = 3, 5 and 8 without n = 40 and T = 8 due to memory problem.

In terms of CPU time, it is quite difficult to obtain the real picture of the performance of the method compared to others because the proposed method is coded in matlab and tested on Intel(R) Core(TM) 2 Duo CPU T7700 @ 2.4 GHz with 2.00 GB of RAM whereas other methods are coded in visual basic program and tested on Pentium 60 and 150 MHz. However, it seems that all the methods require comparably equal CPU time to solve the problem.

For implementing the logic-based model to the DQAP, the proposed methodology provides efficient solutions for such tiny problems by searching all possible permutations. It had taken a lot of computing time but provided poor solutions, when numbers of candidates getting bigger.

Recommendation

The proposed combinatorial method – ABD & ADP with trust-region and successive parameter adaptation procedure – proves to perform comparably well to large-scale DQAP and have a great potential to further improve its performance. The future work toward performance improvement may include:

1. Explore a new combinatorial method based on the proposed method and other metaheuristic methods. The interaction can be either one-way coupling (getting a good initial layout for the other algorithms) or two-way coupling (solve the solution from all methods interactively).

2. Improve accelerating techniques for Benders' decomposition or ABD such as extension of existing trust-region or implication of other techniques.

3. Improve an approximate solution method of large-scale mixed-integer linear programming problem. The method in this study based on linear programming and Hungarian method seems to be part of the reason why the method in this study does not so well perform. Alternative approximate solution based on much more solid ground should significantly improve the proposed combinatorial method.

4. Optimize the successive adaptation procedure in order to gain a better speed performance. Attention should be paid on how to balance trust-region constraint with other parameters in order to allow ABD to generate sample layout containing a true global optimal solution for ADP.

5. To improve the quality of solving the logic-based model, a special technique of searching all-different constraint in ECLiPSe should be studied in case of improving the quality of solutions. Otherwise, benders decomposition, domain reduction techniques and the consistency inference should study further.

LITERATURE CITED

Ahmed, S., M. Goetschalckx, T. Santoso, and A. Shapiro. 2005. A Stochastic Programming Approach to Supply Chain Network Design under Uncertainty. **European Journal of Operational Research**. 167(1): 96-115.

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

_____ and _____. 2000. Genetic Search and the Dynamic Layout Problem. **Computers and Operations Research**. 27: 587-593.

_____, _____ and D.G. Conways. 2000. An Improved Pair-Wise Exchange Heuristic for the Dynamic Plant Layout. **International Journal of Production Research**. 38(13): 3067-3077.

Baykasoğlu, A. and N.N.Z. Gindy. 2001. A Simulated Annealing Algorithm for Dynamic Layout Problem. **Computers and Operations Research**. 28: 403-1426.

Bazaraa, M.S. and H.D. Sherali. 1982. On the Use of Exact and Heuristic Cutting Plane Methods for the Quadratic Assignment Problem. **The Journal of the Operational Research Society** 33: 991-1003.

Benders, J.F. 1962. Partitioning Procedures for Solving Mixed-Variables Programming Problems. **Numerische Mathematik**. 4: 238–252.

Brand, S. 2001. Constraint Propagation in Presence of Arrays. **The 6th Annual Workshop of the ERCIM Working Group on Constraints.** Available Source: http://arxiv.org/abs/cs/0105024v1, January 5, 2009.

Cela, E. 1998. The Quadratic Assignment Problem: Theory and Algorithms. Kluwer Academic Publishers.

Charnsethikul, P. 1999. A Branch and Bound Algorithm for the Dynamic Assignment Problem. **Kasetsart Engineering Journal**. 38: 74-86.

Conway, D.G. and M.A. Venkataramanan. 1994. Genetic Search and the Dynamic Facility Layout Problem. **Computers and Operations Research**. 21: 955-960.

Enscore, E.E. and T.A. Lacksonen. 1993. Quadratic Assignment Algorithms for the Dynamic Layout Problem. **International Journal of Production Research**. 31(3): 503-517.

Heragu, S.S. and J.S. Kochhar. 1999. Facility Layout Design in a Changing Environment. **International Journal of Production Research**. 37(11): 2429-2446.

Hiriart-Urruty, J.B. and C. Lemaréchal. 1996. Convex Analysis and Minimization Algorithms II. **Springer-Verlag**. Berlin Heidelberg.

Hooker, J. 2000. Logic-Based Methods for Optimization Combining Optimization and Constraint Satisfaction. **John Wiley & Sons**. New York.

Lawler, E.L. 1963. The Quadratic Assignment Problem. **Management Science**, 9(4): 586-599.

Luangpaiboon, P. 1995. **Dynamic Process Layout Planning**. M.S. Thesis, Kasetsart University.

Meller, R.D. 1995. **The Facility Layout Problem: A Review of Recent and Emerging Research**. Technical Report 95-03. Department of Industrial Engineering, Auburn University, Alabama.

Muenvanichakul, S. 1998. An Hybrid Approach of Genetic Algorithm/Simulated Annealing and Tabu Search Method for Dynamic Process Layout Planning. Master Thesis, Kasetsart University.

Muenvanichakul, S. 2000. An Hybrid Approach of Genetic Algorithm/Simulated Annealing and Tabu Search Method for Dynamic Process Layout Planning. **In Proceeding of the International Workshop on the Quadratic Assignment Problem and Extension**. 85-94. Kasetsart University, Bangkok, Thailand.

Rosenblatt, M.J. 1986. The Dynamics of Plant Layout. **Management Science**. 32(1): 76-82.

Urban, T.L. 1993. A Heuristic for the Dynamic Facility Layout Problem. **IIE Transaction.** 25(4): 57-63.

_____. 1998. Solution Procedures for the Dynamic Facility Layout Problem. **Annals of Operations Research**. 76: 323-342.

APPENDICES

APPENDIX A

Test Problems

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	36	6	66	12	83	13	58	66	80	60	45	76	80	27	14	11	47	95	47	93	80	11	39	87	23	46	99	42	26	72	36	62	0	80	24	14	48	24	49
2	59	0	49	91	65	22	87	53	55	61	72	39	36	99	17	69	8	48	5	9	66	39	44	33	29	72	31	41	89	85	8	5	30	33	7	49	37	16	68	36
3	94	36	0	82	48	83	40	89	81	50	68	32	65	50	8	72	85	34	68	30	46	48	78	74	81	96	36	6	84	78	28	57	76	4	85	68	96	90	65	85
4	74	84	33	0	69	39	64	68	6	76	46	49	78	3	9	22	73	68	94	7	23	8	94	63	64	63	92	42	74	81	37	23	63	45	97	86	47	33	99	9
5	73	60	26	0	0	54	49	98	69	52	20	51	48	44	71	90	81	88	38	22	30	84	5	34	29	69	75	4	32	81	86	65	23	89	51	53	44	4	41	33
6	37	59	62	87	98	0	67	79	43	34	83	82	98	82	32	51	40	57	35	87	66	31	56	50	8	25	51	92	91	23	82	1	45	95	81	49	75	22	74	51
7	7	28	66	36	57	70	0	40	89	77	22	38	12	76	19	80	84	50	46	16	8	28	80	25	26	63	13	50	5	23	78	81	37	27	66	6	38	43	62	4
8	97	66	36	55	98	92	48	0	25	54	75	99	26	25	84	89	25	92	54	4	50	4	97	14	88	58	55	80	44	89	65	25	60	66	80	9	81	2	41	15
9	5	41	0	52	96	8	10	37	0	47	5	21	38	45	79	9	43	40	81	95	7	61	22	33	88	35	63	1	82	31	69	96	98	50	52	77	36	77	39	71
10	14	85	88	59	45	37	23	46	65	0	51	64	81	72	52	7	9	67	87	10	82	95	7	50	99	33	1	45	41	91	54	0	44	48	72	63	21	80	63	89
11	50	96	55	24	46	30	32	83	93	1	0	38	0	8	84	74	50	4	83	72	97	88	41	10	68	47	62	70	62	70	25	81	18	5	8	0	7	8	91	54
12	20	38	72	15	14	56	3	26	41	73	7	0	31	59	23	86	64	26	81	42	90	50	54	26	64	23	5	5	76	90	55	65	3	92	20	2	83	47	31	72
13	41	68	17	63	14	34	24	69	45	77	44	15	0	22	93	55	32	42	89	82	60	78	89	80	45	65	99	1	38	47	19	48	69	47	46	40	12	42	95	40
14	54	23	98	69	94	37	65	51	31	71	9	7	26	0	4	42	77	65	46	5	70	10	95	68	1	87	38	83	33	76	88	84	42	49	43	29	99	52	76	2
15	96	73	57	8	62	92	89	85	0	27	87	85	62	86	0	93	54	66	78	60	53	50	17	83	69	72	69	57	41	16	68	86	81	54	73	84	41	62	19	73
16	62	22	51	15	11	38	32	30	57	50	59	63	40	91	46	0	50	0	75	53	95	22	82	28	31	3	3	81	30	53	99	65	42	60	26	6	4	44	72	60
17	92	91	17	0	22	3	90	34	82	88	11	32	9	73	23	12	0	3	91	95	91	11	85	65	18	21	67	38	15	11	95	44	84	5	92	90	94	80	83	49
18	80	43	62	37	29	48	26	84	30	67	75	8	43	62	18	54	24	0	84	27	52	4	7	9	4	39	79	62	44	92	25	40	18	68	31	94	59	42	11	28
19	92	47	57	28	85	30	20	89	42	99	97	14	54	20	51	91	62	33	0	9	99	90	88	96	65	35	19	90	90	25	76	24	46	34	27	42	43	30	1	62
20	4	47	0	44	95	93	18	25	60	78	17	15	93	49	89	46	5	48	88	0	14	22	79	39	0	11	33	0	44	91	21	67	21	78	54	68	59	61	23	56
21	65	20	29	57	21	21	86	68	76	17	48	12	85	18	78	57	28	17	86	90	0	53	5	69	63	15	63	1	82	54	59	99	63	26	75	49	3	90	28	86
22	13	83	17	34	18	18	31	50	58	99	27	17	77	40	97	29	49	45	34	76	67	0	28	31	20	70	70	51	29	82	3	50	71	6	79	29	12	16	78	41
23	18	55	6	56	17	89	49	45	56	51	88	39	11	91	86	75	86	41	82	32	33	46	0	89	18	29	53	62	29	35	63	80	38	93	94	31	39	30	57	60
24	96	7	23	48	92	9	46	54	70	9	62	8	72	1	59	21	7	29	83	70	47	20	39	0	46	81	22	33	94	54	33	77	10	61	78	68	14	48	34	5
25	1	81	41	42	7	52	44	72	84	82	13	28	14	90	78	27	8	75	48	9	10	91	95	65	0	53	61	41	91	62	54	60	92	74	53	72	49	27	31	58
26	36	42	36	57	84	2	56	93	81	7	43	1	97	26	82	18	29	97	22	29	9	8	11	97	59	0	47	14	74	43	73	89	94	99	12	57	78	55	79	33
27	75	5	21	47	65	21	91	24	32	70	19	81	78	42	71	23	8	96	82	31	86	54	85	30	97	6	0	90	27	34	26	62	42	66	36	94	55	49	68	72
28	97	96	13	43	75	1	56	68	81	84	5	19	8	59	23	38	23	36	41	40	5	61	45	53	94	36	37	0	41	66	2	22	26	28	69	26	96	34	54	85
29	60	95	48	56	37	78	43	32	5	21	53	18	51	94	32	82	79	22	16	88	28	98	5	6	89	20	52	69	0	41	76	22	11	4	28	5	67	2	70	46
30	78	0	48	92	13	42	76	22	53	94	53	96	19	94	50	33	82	70	9	87	43	1	72	27	8	78	41	25	51	0	44	61	32	96	67	47	36	71	65	28
31	73	89	94	57	86	88	46	1	10	71	94	3	83	97	99	0	41	16	0	10	70	29	22	24	11	52	30	93	27	89	0	18	31	84	0	54	15	99	13	39
32	34	93	61	89	82	34	93	61	89	82	33	23	75	92	86	44	53	65	91	16	76	31	66	77	55	81	57	11	44	11	44	0	61	97	38	5	94	94	75	26
33	74	64	43	63	75	74	20	52	67	28	55	55	99	90	12	34	16	11	93	76	7	62	96	13	22	77	27	36	32	5	71	30	0	54	77	6	1	93	98	34
34	45	40	92	15	55	8	87	5	16	97	46	31	94	63	1	39	95	93	96	57	70	37	84	74	33	3	16	88	91	92	48	97	67	0	17	63	15	46	93	8
35	8	70	0	34	16	0	18	89	32	4	21	24	86	91	92	56	34	75	27	76	67	37	94	40	73	97	48	67	58	94	5	50	19	65	0	54	74	5	59	39
36	27	79	45	3	8	59	11	91	62	44	27	4	84	99	52	1	65	44	77	40	36	53	76	71	81	84	46	57	38	48	53	30	18	5	50	0	80	90	66	27
37	25	55	13	42	25	77	0	87	62	61	73	21	95	26	28	74	17	18	27	97	29	46	63	57	46	82	61	11	37	59	6	65	81	88	10	42	0	45	62	76
38	67	24	3	12	62	11	5	27	97	71	83	0	72	83	76	96	16	59	28	37	88	84	62	93	61	91	24	89	89	64	15	55	57	63	16	0	39	0	54	32
39	42	7	52	36	29	60	2	15	64	3	73	46	56	57	94	58	14	32	9	85	61	77	37	23	86	97	59	70	20	68	35	50	46	5	48	60	41	98	0	54
40	29	74	7	28	4	98	80	37	75	66	31	7	13	69	13	89	69	72	54	92	88	38	95	7	10	98	0	56	53	45	57	84	71	57	68	61	81	37	17	0

Appendix Figure A1 The amount of flow between facility *i* to facility *k* at period 1 (F_{ik1})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	71	72	30	71	32	67	54	22	62	72	56	13	32	90	59	5	87	35	66	1	70	78	97	45	85	8	57	43	99	54	18	32	46	60	93	9	38	14	36
2	2	0	40	90	6	64	84	40	25	13	71	4	47	17	39	71	70	80	7	7	0	57	97	33	1	68	55	7	80	92	5	81	58	16	5	96	61	43	12	42
3	85	6	0	56	58	31	96	11	20	27	65	82	57	2	30	98	49	27	28	43	23	34	85	6	13	20	53	88	32	71	27	58	95	97	29	17	5	70	41	24
4	54	54	26	0	87	87	99	12	82	99	85	70	89	9	82	83	36	24	89	8	1	13	76	80	40	49	99	28	11	12	38	82	85	36	24	26	58	22	89	12
5	22	6	50	55	0	92	75	30	60	44	97	4	85	72	51	59	58	13	24	54	75	25	33	16	68	82	12	83	31	27	87	97	46	88	9	89	71	87	41	73
6	71	10	99	14	5	0	49	82	14	24	36	31	67	20	18	61	62	96	14	11	14	15	99	7	60	57	3	23	71	70	92	80	52	58	29	72	16	94	21	19
7	91	3	24	97	36	61	0	99	41	56	37	78	66	89	59	14	59	5	14	56	29	61	84	17	17	4	5	71	22	28	35	18	68	9	61	43	89	26	25	47
8	19	87	61	9	9	65	70	0	37	2	22	14	40	61	34	92	3	98	31	47	38	46	28	26	81	98	19	31	26	49	26	49	34	75	28	21	28	36	65	44
9	36	87	75	38	89	95	90	6	0	55	53	56	80	9	71	9	93	23	94	9	4	33	25	55	58	79	11	9	84	25	51	92	44	63	41	42	13	98	49	72
10	70	68	62	36	34	43	7	8	94	0	34	76	7	94	85	28	47	69	65	16	58	72	4	37	49	44	10	47	46	44	30	93	78	70	93	63	34	4	34	81
11	42	10	60	63	30	39	99	34	52	16	0	83	79	48	24	52	81	56	15	85	8	63	64	82	90	91	11	29	96	47	78	92	4	32	37	16	91	6	84	43
12	50	48	95	25	29	80	19	28	57	91	70	0	74	99	22	40	16	68	6	18	89	14	65	68	50	83	4	51	25	84	6	56	51	42	38	73	67	44	49	64
13	19	83	31	15	57	54	94	23	34	72	36	62	0	70	81	27	36	53	89	53	10	93	51	49	49	5	41	11	85	22	19	97	7	69	45	9	1	36	30	29
14	18	74	51	43	32	95	77	97	74	52	63	83	39	0	51	50	70	39	39	10	44	60	27	25	33	52	23	69	76	54	79	2	95	59	46	18	51	84	12	56
15	12	4	48	34	51	42	16	20	42	12	1	20	61	91	0	76	27	61	66	24	8	96	90	38	8	4	34	7	13	82	56	50	5	17	72	38	66	48	50	41
16	95	10	91	94	74	95	58	17	75	52	85	59	47	33	95	0	38	99	85	13	47	64	80	57	86	66	91	1	21	52	43	31	98	12	37	43	62	77	7	52
17	32	81	33	19	20	0	74	91	57	71	64	92	12	99	47	67	0	61	86	50	29	29	68	61	24	12	89	93	20	11	17	86	39	31	74	97	79	19	30	33
18	20	55	74	43	69	86	0	86	90	7	97	67	39	55	98	62	53	0	0	51	23	75	64	39	55	87	92	53	8	59	66	21	89	65	2	80	53	45	55	13
19	28	64	98	84	94	29	79	41	87	9	34	45	42	65	10	95	91	27	0	92	7	59	72	7	58	70	30	39	18	40	14	98	32	92	74	40	34	75	15	92
20	70	16	94	11	11	21	8	4	58	6	26	70	17	87	48	35	8	86	97	0	83	95	2	47	99	84	9	84	81	80	91	75	99	15	6	70	51	87	60	31
21	31	36	89	91	20	5	83	68	90	58	17	45	60	66	39	61	93	63	21	58	0	39	18	4	55	19	38	14	41	8	66	54	91	45	49	2	45	28	88	86
22	3	17	79	53	29	34	94	55	16	17	52	65	28	9	10	26	92	45	19	54	2	0	78	18	36	56	41	37	52	39	43	23	6	41	61	60	18	14	34	7
23	35	27	99	80	65	51	54	44	24	37	73	6	98	36	80	7	32	77	22	25	46	7	0	5	71	42	77	43	59	85	58	48	61	14	3	86	80	81	43	84
24	28	55	83	17	10	66	27	92	12	8	5	35	53	82	94	49	10	66	82	50	41	49	0	0	81	85	85	75	0	44	45	58	0	37	26	31	17	54	27	40
25	14	93	4	49	14	30	96	46	87	63	20	81	5	26	81	54	76	39	41	6	43	11	18	93	0	32	1	77	82	50	63	80	87	30	46	2	92	69	83	29
26	93	26	77	0	28	4	98	33	76	53	3	93	37	8	46	42	43	60	74	54	66	77	28	79	82	0	74	68	65	14	81	0	54	40	30	92	98	5	41	96
27	99	84	1	72	32	21	92	27	73	57	2	43	61	16	18	85	46	5	23	62	19	94	89	18	99	31	0	35	91	69	25	70	54	62	13	13	74	89	18	89
28	64	7	83	96	44	9	73	10	99	21	64	3	5	60	85	41	19	33	43	57	8	20	31	38	82	20	8	0	14	33	34	27	59	39	5	80	55	18	3	93
29	30	2	34	28	80	3	93	94	40	47	65	97	68	62	90	26	0	88	4	49	10	32	29	95	87	97	42	35	0	99	95	43	24	99	99	39	40	93	0	87
30	67	76	13	65	55	91	36	64	41	66	52	4	90	76	65	58	6	18	30	5	71	77	92	70	33	37	15	41	20	0	79	49	30	3	30	8	27	50	96	62
31	2	9	54	67	2	31	66	20	49	22	51	67	57	36	5	38	98	26	23	31	82	19	50	28	24	17	13	39	93	34	0	50	71	7	20	40	76	96	84	27
32	66	82	0	32	25	66	82	0	32	25	41	16	35	93	83	24	74	4	88	30	87	18	5	34	14	46	63	11	57	11	57	0	36	54	29	31	99	85	78	17
33	50	89	61	78	57	78	52	9	73	7	48	43	96	43	89	57	45	82	53	34	61	11	66	14	84	1	1	87	20	37	60	0	0	56	65	44	90	44	26	11
34	23	60	81	72	4	74	85	23	47	29	11	14	35	87	67	34	7	87	85	28	97	46	48	53	3	24	48	14	91	78	83	84	23	0	84	1	2	66	28	36
35	52	0	57	86	33	61	10	51	55	31	95	78	17	0	43	48	48	94	75	37	72	80	36	69	59	7	4	84	47	66	82	53	55	94	0	93	49	48	17	49
36	42	12	88	19	12	43	12	97	14	5	93	91	21	2	53	6	49	59	81	31	4	36	54	3	31	93	64	31	67	21	32	80	38	66	13	0	68	64	41	97
37	16	37	37	1	61	17	68	8	68	26	42	86	28	45	99	33	19	83	71	34	0	77	18	91	51	90	31	50	74	23	81	50	9	85	59	32	0	65	32	3
38	65	3	44	62	26	8	34	31	58	70	73	61	9	98	97	71	18	1	33	0	89	98	0	33	70	96	7	69	30	12	74	59	82	74	27	15	13	0	14	6
39	62	66	45	0	72	68	59	8	52	19	47	33	17	57	50	55	84	7	76	57	66	61	97	76	19	57	93	63	45	67	67	46	51	1	77	4	27	30	0	80
40	31	35	15	18	74	94	57	21	5	56	19	71	17	70	30	32	97	96	59	45	69	76	46	50	59	59	28	64	48	81	88	19	41	5	81	12	72	96	65	0

Appendix Figure A2 The amount of flow between facility *i* to facility *k* at period 2 (F_{ik2})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	12	15	9	0	71	5	37	53	97	44	16	80	26	72	99	61	74	42	26	32	90	10	18	11	49	84	3	8	31	53	92	22	30	97	73	10	88	72	42
2	39	0	4	43	94	22	38	87	33	94	95	43	96	11	75	69	81	57	48	62	89	33	80	12	28	89	14	85	65	30	13	96	9	96	35	57	21	31	39	75
3	54	32	0	3	77	55	0	82	10	95	11	27	13	43	84	50	10	71	79	2	71	37	29	18	9	43	28	50	66	56	82	18	66	50	23	1	60	91	28	36
4	23	25	39	0	79	29	5	30	57	53	89	76	19	52	35	59	23	68	25	2	91	49	22	6	33	92	57	70	63	59	60	25	19	82	23	50	24	40	95	69
5	10	91	81	67	0	80	55	27	96	13	33	93	26	34	46	54	73	50	41	17	71	87	28	45	87	52	86	45	18	81	75	67	62	76	25	37	76	75	7	11
6	11	44	93	95	24	0	79	64	91	26	93	84	46	60	28	25	57	40	44	73	5	81	36	17	84	71	98	77	38	44	74	80	97	22	33	4	28	3	41	68
7	91	20	89	78	31	13	0	94	82	9	94	77	49	0	79	5	5	37	79	18	88	73	1	82	66	7	78	2	89	17	21	34	99	7	33	4	95	51	81	40
8	74	45	45	87	30	30	79	0	74	4	34	62	32	66	89	27	29	11	74	76	37	51	38	92	20	13	54	73	49	76	12	14	21	55	11	54	80	49	63	64
9	67	53	19	69	37	87	97	84	0	6	11	39	23	82	69	25	13	41	69	33	25	25	45	56	69	87	76	15	76	58	48	60	55	21	70	63	21	80	42	25
10	3	14	52	62	58	38	49	95	52	0	26	39	16	44	67	50	98	91	57	57	80	72	73	60	7	82	49	24	62	32	47	41	16	34	25	24	80	53	24	55
11	75	87	89	37	3	0	20	77	49	21	0	39	18	38	17	38	65	31	61	98	7	3	60	84	97	90	83	56	74	2	38	51	47	41	97	77	40	49	36	19
12	80	79	12	43	65	69	23	22	33	48	95	39	59	35	71	28	9	37	43	19	43	82	11	47	90	30	79	75	72	94	95	16	86	44	89	76	68	16	55	9
13	21	20	91	93	43	43	81	26	43	81	79	39	0	17	99	72	93	93	7	44	32	52	56	27	7	99	69	36	20	98	60	84	86	71	15	19	49	90	22	45
14	41	64	54	98	88	90	71	36	27	79	85	39	49	0	31	31	77	73	85	53	51	70	76	6	37	82	27	22	19	50	88	2	84	35	42	49	55	55	60	55
15	62	36	50	0	45	5	22	60	64	74	30	39	64	64	0	78	86	49	63	67	32	44	92	22	89	62	23	79	53	86	42	2	58	58	53	36	49	78	95	43
16	15	93	47	90	65	11	19	86	87	77	69	39	93	36	97	0	68	91	58	6	72	69	25	37	76	75	3	77	48	42	50	97	41	78	61	20	16	20	98	56
17	98	93	67	19	32	7	47	5	30	74	8	39	57	98	52	51	0	15	67	95	79	31	11	43	5	73	21	30	49	44	76	84	25	11	21	97	29	99	78	83
18	77	97	11	44	5	17	52	64	88	82	92	39	95	89	12	31	26	0	44	80	72	80	71	41	66	76	63	7	14	79	59	58	32	67	16	7	94	80	45	9
19	16	76	22	71	19	88	1	24	61	35	38	39	12	58	98	74	1	83	0	67	6	48	73	96	87	86	95	74	47	65	57	55	5	44	47	45	31	65	24	4
20	89	71	77	76	64	35	39	15	28	96	11	39	25	48	23	75	21	9	9	0	67	81	51	65	87	32	67	36	1	92	91	49	2	84	40	12	0	27	64	14
21	9	51	14	7	20	45	56	84	57	88	75	39	78	20	94	60	79	82	19	29	0	60	47	37	65	9	48	56	97	15	40	67	85	95	57	48	82	38	14	82
22	48	20	96	30	76	62	28	88	48	51	52	39	33	41	50	40	64	79	16	98	11	0	80	92	29	67	22	91	29	83	82	73	38	55	96	99	83	22	30	37
23	70	20	28	0	39	67	59	62	74	93	21	39	96	40	95	51	24	63	78	26	71	6	0	8	82	11	43	63	78	68	50	31	49	9	29	80	68	28	75	2
24	34	81	86	87	61	58	4	18	38	80	17	39	27	1	71	12	6	32	28	4	16	11	88	0	18	42	3	82	15	45	22	22	63	88	83	30	10	55	83	3
25	87	17	11	57	45	16	9	97	44	93	66	39	92	17	51	23	98	6	92	65	20	34	67	91	0	13	37	80	55	83	21	83	32	72	99	13	86	83	2	47
26	89	37	9	59	57	1	43	16	77	78	36	39	24	21	80	86	21	91	89	22	81	11	86	77	73	0	63	84	19	99	79	77	63	91	89	34	33	29	50	11
27	33	6	73	80	85	83	34	77	9	84	94	39	89	6	38	32	46	20	14	62	0	21	55	54	58	5	0	46	5	85	41	69	54	72	54	27	55	12	34	33
28	40	75	73	8	88	40	87	45	69	35	7	39	14	91	48	35	69	19	64	46	90	39	75	60	87	80	49	0	75	0	88	67	62	90	13	7	55	97	3	3
29	10	13	32	76	52	89	52	96	75	41	74	39	3	40	26	94	54	29	46	97	47	61	25	94	16	32	23	52	0	0	38	75	64	86	33	37	43	99	99	49
30	34	14	0	60	68	46	51	93	20	58	7	39	89	32	38	58	78	56	67	32	11	56	89	92	63	36	62	73	37	0	72	78	57	40	14	45	57	58	41	21
31	90	13	92	60	2	24	21	82	54	5	34	39	44	3	28	40	61	4	17	30	60	1	34	51	44	29	42	78	59	47	0	58	92	35	12	85	23	36	5	52
32	38	99	96	44	76	38	99	96	44	76	80	39	50	10	13	29	51	87	42	14	33	91	83	47	25	9	84	94	61	94	61	0	44	85	61	37	18	63	46	18
33	69	13	36	91	62	83	98	14	40	56	96	39	60	44	82	94	45	66	4	87	40	74	35	57	71	31	13	76	18	54	91	44	0	32	16	50	21	62	36	30
34	12	12	56	74	16	0	16	10	52	53	15	39	67	21	22	99	37	61	85	59	64	45	86	90	30	74	56	42	18	98	33	65	72	0	51	66	91	73	58	16
35	21	68	44	62	24	74	38	48	60	56	5	39	85	76	13	1	75	37	57	18	9	6	65	30	59	6	16	66	86	50	68	41	41	44	0	30	11	36	82	9
36	8	48	27	19	83	5	84	6	75	94	81	39	9	28	91	89	28	95	76	90	83	56	64	40	65	24	74	26	60	19	56	54	83	77	62	0	45	64	47	48
37	37	81	47	68	10	68	15	45	61	11	17	39	69	33	97	6	11	34	35	58	33	50	87	88	5	48	99	89	56	43	12	2	40	70	4	82	0	62	16	24
38	24	60	51	2	82	24	88	92	29	96	38	39	83	97	35	78	30	95	80	15	62	17	21	18	38	42	58	41	86	78	83	37	83	3	84	37	75	0	66	76
39	79	25	99	86	67	16	0	43	31	33	58	39	41	50	90	60	23	90	82	31	49	75	87	75	2	8	9	79	30	86	73	17	71	47	91	66	28	60	0	70
40	32	2	56	35	37	34	30	41	84	35	7	39	85	77	88	82	79	97	9	61	58	71	62	35	26	78	36	61	85	34	4	2	58	20	84	52	74	35	71	0

Appendix Figure A3 The amount of flow between facility *i* to facility *k* at period 3 (F_{ik3})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	14	20	36	83	21	21	1	67	12	71	74	34	3	13	38	77	12	7	67	82	71	60	72	65	84	19	91	94	20	68	23	9	53	26	34	53	66	61	4
2	32	0	96	38	4	78	31	26	6	7	61	55	87	11	23	75	93	65	87	98	36	57	46	93	72	76	27	81	66	88	1	28	8	5	61	59	90	79	47	91
3	71	97	0	24	89	39	31	42	36	3	41	49	87	97	56	10	18	78	89	51	5	58	21	63	92	85	84	91	32	37	33	23	44	69	30	47	70	17	29	75
4	28	16	26	0	85	8	78	17	36	25	32	84	35	54	25	15	89	3	7	75	41	23	89	68	97	16	78	49	75	86	51	49	26	93	70	11	15	12	74	28
5	56	13	97	37	0	76	73	11	57	84	62	16	93	75	74	22	26	20	53	33	60	57	74	48	28	43	1	58	40	20	89	48	12	17	90	45	5	2	73	29
6	98	74	57	8	50	0	86	48	76	83	68	78	0	31	71	55	32	21	15	82	17	98	56	10	88	38	44	1	14	77	13	67	79	13	3	91	92	92	6	33
7	24	14	46	75	4	76	0	22	47	30	85	92	12	37	73	61	69	42	83	82	65	82	28	37	36	16	73	57	14	92	61	83	2	41	1	46	39	2	51	86
8	53	47	80	16	89	46	88	0	15	46	15	20	1	2	1	17	14	2	96	50	67	37	75	90	26	4	81	34	49	95	57	64	56	67	2	81	28	99	69	28
9	68	13	35	6	73	96	8	16	0	90	8	0	38	49	19	73	16	91	74	22	65	34	47	16	30	79	75	19	75	17	89	38	29	65	26	93	14	31	22	31
10	63	22	18	33	47	35	67	79	6	0	67	8	68	38	8	89	19	13	53	68	15	61	73	66	64	31	34	40	53	75	24	11	61	36	2	65	0	17	81	39
11	72	32	62	34	94	75	13	69	91	17	0	94	43	52	64	52	34	38	70	85	54	42	72	10	23	67	35	36	38	99	6	95	73	13	11	90	16	59	36	9
12	83	59	26	26	9	77	44	59	21	26	13	0	85	70	83	38	50	92	38	40	66	58	92	86	35	51	4	87	77	19	84	52	97	20	35	68	37	79	50	68
13	26	69	38	91	59	82	64	9	11	58	30	22	0	26	67	5	14	69	55	7	44	42	22	11	57	65	27	86	60	64	91	75	79	2	89	22	12	34	83	90
14	85	98	95	88	60	23	64	18	53	67	74	2	11	0	17	65	66	81	9	21	67	52	15	3	27	92	52	54	55	55	92	99	38	85	93	44	50	59	99	46
15	39	44	29	81	62	73	14	18	4	23	32	91	27	83	0	50	89	2	26	23	21	79	42	87	97	65	75	20	37	10	89	80	41	92	34	22	84	15	88	15
16	54	17	54	61	69	10	77	17	29	75	83	74	73	27	84	0	3	64	39	92	98	30	14	72	57	18	61	24	28	97	18	35	32	57	56	70	11	74	45	13
17	97	91	70	35	56	31	32	10	32	33	12	56	9	40	19	43	0	2	5	26	65	56	57	64	95	45	58	63	83	39	55	95	53	19	6	12	37	14	14	47
18	33	45	45	54	21	11	27	4	61	32	30	17	99	70	96	40	48	0	25	2	60	94	20	80	27	56	0	56	45	5	60	80	25	46	34	42	72	41	72	10
19	36	20	91	68	95	96	53	24	96	58	99	13	0	39	56	35	12	90	0	72	77	52	86	78	87	70	34	90	94	22	26	73	96	99	6	31	62	39	99	73
20	25	39	15	83	24	68	39	4	84	92	0	3	69	76	6	3	2	30	13	0	45	20	2	90	8	79	13	48	34	72	97	32	38	92	82	66	19	18	20	43
21	1	87	92	35	47	89	56	88	70	13	82	23	55	63	25	76	18	78	61	83	0	13	43	37	69	87	32	22	37	13	28	45	19	31	83	38	85	56	48	65
22	71	87	51	31	77	35	46	52	14	11	79	97	94	60	44	8	55	57	11	27	99	0	21	81	11	40	63	18	10	81	78	35	27	94	77	16	17	26	40	95
23	59	36	37	69	46	80	29	16	71	9	27	97	40	68	16	98	99	62	79	33	24	2	0	51	49	85	55	91	84	4	41	86	7	77	5	93	15	91	71	37
24	98	85	13	99	67	68	64	0	18	18	80	93	49	86	40	69	91	21	16	30	41	44	94	0	45	61	6	67	41	36	64	79	55	58	24	10	2	15	23	40
25	75	76	62	79	68	77	43	62	30	74	6	83	1	85	55	2	14	90	94	96	51	22	45	88	0	54	84	33	18	48	39	36	26	36	24	11	96	53	40	74
26	79	14	60	97	5	33	16	83	40	26	7	34	11	39	25	27	58	56	16	2	81	93	23	80	23	0	58	18	64	64	11	34	68	97	86	86	80	16	17	78
27	70	68	97	62	27	67	38	97	21	82	62	85	16	57	35	62	27	1	46	10	50	20	7	23	55	5	0	65	25	50	55	45	56	28	63	45	35	62	47	64
28	71	40	27	42	40	56	70	55	83	14	1	96	3	7	11	98	81	4	88	70	85	5	5	85	85	92	65	0	17	72	4	22	66	87	34	36	27	55	36	64
29	55	2	96	11	48	88	88	18	1	24	93	14	63	20	50	65	20	30	63	7	20	63	38	38	15	14	20	36	0	31	41	73	21	6	80	13	83	4	99	33
30	25	35	68	30	64	55	56	85	59	48	57	27	95	40	26	90	53	17	82	7	93	36	69	63	77	91	47	16	53	0	0	94	77	64	51	31	16	85	62	74
31	46	62	84	61	42	55	37	3	27	21	62	71	8	53	42	5	16	26	31	70	80	95	58	56	21	65	71	76	83	53	0	9	28	96	10	22	83	57	84	41
32	11	41	74	42	54	11	41	74	42	54	68	5	25	77	97	8	75	51	86	17	35	54	80	86	89	81	45	34	13	34	13	0	30	47	74	61	25	92	69	66
33	19	19	31	36	4	11	16	26	1	92	37	13	28	84	74	34	85	76	85	23	3	82	54	37	42	81	70	62	4	42	40	0	0	25	46	85	85	50	49	33
34	11	82	53	20	42	52	92	20	32	44	7	96	78	30	22	12	13	5	19	0	47	4	38	17	75	60	59	45	78	89	62	16	0	0	40	36	62	27	28	85
35	39	31	14	27	51	16	5	6	29	79	36	84	88	5	38	61	3	95	34	63	9	46	80	23	29	0	34	59	36	90	59	22	41	50	0	64	73	7	89	5
36	29	14	18	4	70	32	80	49	29	10	22	31	3	2	62	15	46	15	49	74	48	47	92	58	77	33	57	66	73	97	4	35	94	17	80	0	89	35	58	31
37	64	27	1	23	47	51	77	30	69	74	12	51	94	34	15	89	29	30	31	55	32	30	81	83	38	39	10	95	86	20	47	74	76	25	99	95	0	71	74	32
38	78	39	80	63	15	64	71	40	11	9	69	68	57	88	13	58	38	76	17	42	12	23	64	63	60	82	45	81	77	39	62	81	87	21	29	38	68	0	67	63
39	89	85	6	41	71	0	47	62	86	23	53	34	10	11	53	58	9	68	59	23	24	10	33	16	73	80	68	98	58	28	66	31	73	6	72	70	8	76	0	81
40	55	69	48	97	90	25	58	91	53	42	16	96	4	76	81	23	21	29	1	64	37	90	5	63	32	93	56	98	89	71	46	64	95	25	53	36	24	23	14	0

Appendix Figure A4 The amount of flow between facility *i* to facility *k* at period 4 (F_{ik4})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	8	41	27	82	41	31	69	81	51	13	62	92	11	47	9	55	70	76	55	66	62	38	96	40	40	92	38	75	49	81	32	59	80	24	95	2	0	40	5
2	41	0	6	47	50	96	49	27	95	27	1	52	27	84	21	82	84	18	92	15	48	26	50	72	47	15	50	37	94	45	78	38	24	92	44	52	96	47	60	70
3	33	84	0	49	39	84	8	12	51	56	72	94	80	36	58	53	97	62	90	13	3	82	93	93	3	72	36	1	18	25	81	35	29	19	73	67	48	64	45	60
4	53	69	67	0	8	53	42	40	30	35	6	59	38	32	21	36	77	73	7	55	94	81	2	65	60	5	26	97	45	41	65	82	66	94	25	99	8	67	75	56
5	23	83	55	50	0	82	91	24	34	39	63	70	54	60	26	57	92	64	86	48	6	27	80	78	20	7	98	68	62	98	67	44	94	63	81	38	83	49	52	68
6	39	52	63	4	87	0	85	82	82	65	11	95	23	0	99	77	31	72	68	30	95	85	15	90	57	89	7	86	62	69	33	92	60	22	9	96	66	58	44	68
7	56	55	12	82	62	83	0	94	40	53	77	59	45	58	0	17	62	52	81	16	65	95	56	38	55	38	67	25	10	48	89	55	54	90	76	81	21	66	0	16
8	91	72	80	37	97	36	79	0	8	46	13	93	91	94	46	19	32	61	48	95	42	95	76	27	38	59	1	53	53	39	67	78	82	62	96	20	5	14	53	55
9	25	97	62	30	86	14	72	46	0	86	19	90	7	6	45	44	54	76	14	46	35	65	99	66	46	10	52	84	95	50	24	13	90	13	74	92	40	53	99	31
10	94	38	59	60	44	46	0	69	42	0	76	4	84	31	0	3	38	2	23	16	4	49	54	77	21	3	2	25	61	14	85	61	89	44	85	24	81	69	79	66
11	37	5	97	45	97	52	50	61	55	33	0	74	56	87	21	22	98	20	63	36	56	98	98	6	66	57	79	85	54	69	17	94	59	89	59	58	31	83	52	75
12	73	91	27	24	48	96	8	19	87	3	15	0	70	79	69	96	91	16	33	9	39	98	29	8	31	24	48	82	45	57	62	72	44	34	53	50	92	1	54	65
13	90	31	91	37	72	8	59	47	2	43	86	21	0	71	56	48	24	23	40	92	67	3	18	98	5	81	66	23	44	9	45	84	83	23	73	92	38	89	39	74
14	33	37	79	89	70	98	89	91	68	14	31	84	74	0	7	59	71	72	34	99	87	68	55	83	12	22	99	67	94	3	68	84	20	94	68	94	84	49	71	46
15	69	41	3	86	82	43	92	28	11	27	95	38	2	33	0	81	43	90	59	13	9	19	75	63	0	40	21	21	55	77	74	90	44	73	8	94	9	10	73	47
16	68	85	61	70	77	84	60	5	95	62	65	78	31	33	22	0	77	15	92	93	49	10	64	58	38	98	11	57	55	79	38	28	27	70	25	2	63	93	77	49
17	53	89	37	91	33	23	10	19	72	66	90	83	39	80	10	3	0	34	32	56	21	83	94	29	84	50	97	48	53	52	87	59	56	17	85	71	52	15	7	13
18	79	99	4	57	29	86	49	40	54	89	91	18	31	80	1	40	53	0	18	49	75	56	88	4	53	44	79	39	95	77	18	83	88	99	70	11	76	10	71	48
19	65	63	4	39	3	10	86	35	1	67	38	70	93	98	29	51	34	18	0	44	19	65	26	43	59	25	49	23	57	18	14	40	17	27	0	19	52	52	29	32
20	91	90	65	72	56	44	4	62	93	6	63	81	17	50	73	10	47	98	85	0	50	44	59	7	52	68	57	41	14	40	88	10	95	27	67	78	22	33	46	82
21	87	69	45	9	84	60	67	94	68	65	30	86	30	42	43	96	71	9	71	89	0	20	93	67	70	96	74	69	96	13	60	62	13	65	11	33	27	11	4	90
22	28	58	1	37	94	21	58	48	78	8	91	85	53	95	7	20	84	39	39	15	8	0	0	44	29	67	64	83	60	60	14	56	74	85	33	21	2	62	55	80
23	61	81	94	98	55	72	96	8	59	63	92	66	73	45	34	16	6	0	25	87	66	69	0	42	34	70	59	9	55	2	66	66	37	18	15	70	25	70	40	38
24	12	40	35	65	52	44	30	28	87	40	77	85	74	70	39	56	29	89	65	62	75	17	87	0	50	76	97	72	13	50	2	69	1	57	7	93	87	82	53	15
25	25	20	81	43	34	18	47	24	79	36	78	82	99	85	64	58	11	75	95	35	34	75	16	15	0	19	45	7	24	15	81	85	69	74	15	0	34	14	42	79
26	0	57	27	71	6	68	4	17	22	65	5	99	31	17	86	42	53	23	99	93	19	47	76	99	50	0	14	85	94	20	99	57	3	83	11	63	43	33	74	60
27	11	6	97	10	25	0	75	74	41	7	22	26	95	42	91	66	21	60	18	40	13	95	30	17	78	43	0	26	66	98	33	18	90	48	5	47	30	66	68	57
28	64	12	6	69	8	54	26	16	2	10	35	67	67	34	14	17	5	20	19	64	20	8	94	2	61	68	70	0	70	57	19	6	10	43	64	34	53	96	26	2
29	19	43	92	28	47	19	52	14	9	56	31	71	31	14	4	95	40	52	99	0	42	75	91	97	28	86	59	13	0	52	45	2	63	69	93	47	93	16	24	92
30	13	68	81	6	89	43	38	75	33	64	85	3	77	8	61	25	84	52	36	83	65	43	22	68	14	38	74	38	59	0	60	18	45	95	45	24	50	14	98	63
31	20	75	16	26	51	41	10	26	43	39	88	7	92	91	94	58	1	17	62	8	12	69	59	24	11	51	91	25	77	62	0	83	77	25	77	9	90	10	96	59
32	50	93	35	92	97	50	93	35	92	97	88	24	63	78	18	94	15	44	98	66	50	46	49	94	6	61	14	89	47	89	47	0	48	59	32	6	55	57	32	62
33	21	14	69	84	64	21	61	4	91	70	39	98	8	20	94	27	97	47	18	33	20	91	65	13	58	19	41	11	27	90	74	7	0	17	54	6	79	47	76	68
34	97	16	25	90	21	31	76	55	66	60	48	84	91	87	13	65	8	85	94	78	7	38	25	22	97	23	12	98	37	40	97	45	87	0	57	12	67	13	35	73
35	34	21	81	30	67	52	48	24	0	59	69	99	47	82	17	17	94	89	78	27	92	9	52	86	19	66	10	65	82	26	6	74	91	40	0	65	72	0	69	27
36	53	6	42	71	19	39	68	54	30	64	28	9	87	5	29	1	39	40	27	41	30	76	33	14	11	63	98	24	54	34	54	92	92	37	51	0	99	58	42	85
37	65	20	91	40	53	60	52	74	78	40	14	33	99	16	35	63	46	28	31	22	70	78	64	77	29	96	70	85	6	83	21	77	70	16	61	34	0	97	39	59
38	66	72	0	86	29	7	22	21	34	87	70	86	57	66	6	10	95	56	25	33	14	95	32	69	96	51	89	2	50	52	58	0	36	24	27	20	8	0	30	10
39	34	26	12	59	43	85	5	58	22	55	79	31	17	34	6	98	32	77	64	66	80	45	72	36	56	35	56	1	44	82	47	18	58	7	35	37	57	4	0	16
40	76	80	44	29	57	66	24	7	78	60	24	56	95	29	65	73	46	34	61	38	73	44	9	72	22	75	56	12	31	68	49	29	92	90	46	86	72	4	17	0

Appendix Figure A5 The amount of flow between facility *i* to facility *k* at period 5 (F_{ik5})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	71	72	30	71	32	67	54	22	62	72	56	13	32	90	59	5	87	35	66	1	70	78	97	45	85	8	57	43	99	54	18	32	46	60	93	9	38	14	36
2	2	0	40	90	6	64	84	40	25	13	71	4	47	17	39	71	70	80	7	7	0	57	97	33	1	68	55	7	80	92	5	81	58	16	5	96	61	43	12	42
3	85	6	0	56	58	31	96	11	20	27	65	82	57	2	30	98	49	27	28	43	23	34	85	6	13	20	53	88	32	71	27	58	95	97	29	17	5	70	41	24
4	54	54	26	0	87	87	99	12	82	99	85	70	89	9	82	83	36	24	89	8	1	13	76	80	40	49	99	28	11	12	38	82	85	36	24	26	58	22	89	12
5	22	6	50	55	0	92	75	30	60	44	97	4	85	72	51	59	58	13	24	54	75	25	33	16	68	82	12	83	31	27	87	97	46	88	9	89	71	87	41	73
6	71	10	99	14	5	0	49	82	14	24	36	31	67	20	18	61	62	96	14	11	14	15	99	7	60	57	3	23	71	70	92	80	52	58	29	72	16	94	21	19
7	91	3	24	97	36	61	0	99	41	56	37	78	66	89	59	14	59	5	14	56	29	61	84	17	17	4	5	71	22	28	35	18	68	9	61	43	89	26	25	47
8	19	87	61	9	9	65	70	0	37	2	22	14	40	61	34	92	3	98	31	47	38	46	28	26	81	98	19	31	26	49	26	49	34	75	28	21	28	36	65	44
9	36	87	75	38	89	95	90	6	0	55	53	56	80	9	71	9	93	23	94	9	4	33	25	55	58	79	11	9	84	25	51	92	44	63	41	42	13	98	49	72
10	70	68	62	36	34	43	7	8	94	0	34	76	7	94	85	28	47	69	65	16	58	72	4	37	49	44	10	47	46	44	30	93	78	70	93	63	34	4	34	81
11	42	10	60	63	30	39	99	34	52	16	0	83	79	48	24	52	81	56	15	85	8	63	64	82	90	91	11	29	96	47	78	92	4	32	37	16	91	6	84	43
12	50	48	95	25	29	80	19	28	57	91	70	0	74	99	22	40	16	68	6	18	89	14	65	68	50	83	4	51	25	84	6	56	51	42	38	73	67	44	49	64
13	19	83	31	15	57	54	94	23	34	72	36	62	0	70	81	27	36	53	89	53	10	93	51	49	49	5	41	11	85	22	19	97	7	69	45	9	1	36	30	29
14	18	74	51	43	32	95	77	97	74	52	63	83	39	0	51	50	70	39	39	10	44	60	27	25	33	52	23	69	76	54	79	2	95	59	46	18	51	84	12	56
15	12	4	48	34	51	42	16	20	42	12	1	20	61	91	0	76	27	61	66	24	8	96	90	38	8	4	34	7	13	82	56	50	5	17	72	38	66	48	50	41
16	95	10	91	94	74	95	58	17	75	52	85	59	47	33	95	0	38	99	85	13	47	64	80	57	86	66	91	1	21	52	43	31	98	12	37	43	62	77	7	52
17	32	81	33	19	20	0	74	91	57	71	64	92	12	99	47	67	0	61	86	50	29	29	68	61	24	12	89	93	20	11	17	86	39	31	74	97	79	19	30	33
18	20	55	74	43	69	86	0	86	90	7	97	67	39	55	98	62	53	0	0	51	23	75	64	39	55	87	92	53	8	59	66	21	89	65	2	80	53	45	55	13
19	28	64	98	84	94	29	79	41	87	9	34	45	42	65	10	95	91	27	0	92	7	59	72	7	58	70	30	39	18	40	14	98	32	92	74	40	34	75	15	92
20	70	16	94	11	11	21	8	4	58	6	26	70	17	87	48	35	8	86	97	0	83	95	2	47	99	84	9	84	81	80	91	75	99	15	6	70	51	87	60	31
21	31	36	89	91	20	5	83	68	90	58	17	45	60	66	39	61	93	63	21	58	0	39	18	4	55	19	38	14	41	8	66	54	91	45	49	2	45	28	88	86
22	3	17	79	53	29	34	94	55	16	17	52	65	28	9	10	26	92	45	19	54	2	0	78	18	36	56	41	37	52	39	43	23	6	41	61	60	18	14	34	7
23	35	27	99	80	65	51	54	44	24	37	73	6	98	36	80	7	32	77	22	25	46	7	0	5	71	42	77	43	59	85	58	48	61	14	3	86	80	81	43	84
24	28	55	83	17	10	66	27	92	12	8	5	35	53	82	94	49	10	66	82	50	41	49	0	0	81	85	85	75	0	44	45	58	0	37	26	31	17	54	27	40
25	14	93	4	49	14	30	96	46	87	63	20	81	5	26	81	54	76	39	41	6	43	11	18	93	0	32	1	77	82	50	63	80	87	30	46	2	92	69	83	29
26	93	26	77	0	28	4	98	33	76	53	3	93	37	8	46	42	43	60	74	54	66	77	28	79	82	0	74	68	65	14	81	0	54	40	30	92	98	5	41	96
27	99	84	1	72	32	21	92	27	73	57	2	43	61	16	18	85	46	5	23	62	19	94	89	18	99	31	0	35	91	69	25	70	54	62	13	13	74	89	18	89
28	64	7	83	96	44	9	73	10	99	21	64	3	5	60	85	41	19	33	43	57	8	20	31	38	82	20	8	0	14	33	34	27	59	39	5	80	55	18	3	93
29	30	2	34	28	80	3	93	94	40	47	65	97	68	62	90	26	0	88	4	49	10	32	29	95	87	97	42	35	0	99	95	43	24	99	99	39	40	93	0	87
30	67	76	13	65	55	91	36	64	41	66	52	4	90	76	65	58	6	18	30	5	71	77	92	70	33	37	15	41	20	0	79	49	30	3	30	8	27	50	96	62
31	2	9	54	67	2	31	66	20	49	22	51	67	57	36	5	38	98	26	23	31	82	19	50	28	24	17	13	39	93	34	0	50	71	7	20	40	76	96	84	27
32	66	82	0	32	25	66	82	0	32	25	41	16	35	93	83	24	74	4	88	30	87	18	5	34	14	46	63	11	57	11	57	0	36	54	29	31	99	85	78	17
33	50	89	61	78	57	78	52	9	73	7	48	43	96	43	89	57	45	82	53	34	61	11	66	14	84	1	1	87	20	37	60	0	0	56	65	44	90	44	26	11
34	23	60	81	72	4	74	85	23	47	29	11	14	35	87	67	34	7	87	85	28	97	46	48	53	3	24	48	14	91	78	83	84	23	0	84	1	2	66	28	36
35	52	0	57	86	33	61	10	51	55	31	95	78	17	0	43	48	48	94	75	37	72	80	36	69	59	7	4	84	47	66	82	53	55	94	0	93	49	48	17	49
36	42	12	88	19	12	43	12	97	14	5	93	91	21	2	53	6	49	59	81	31	4	36	54	3	31	93	64	31	67	21	32	80	38	66	13	0	68	64	41	97
37	16	37	37	1	61	17	68	8	68	26	42	86	28	45	99	33	19	83	71	34	0	77	18	91	51	90	31	50	74	23	81	50	9	85	59	32	0	65	32	3
38	65	3	44	62	26	8	34	31	58	70	73	61	9	98	97	71	18	1	33	0	89	98	0	33	70	96	7	69	30	12	74	59	82	74	27	15	13	0	14	6
39	62	66	45	0	72	68	59	8	52	19	47	33	17	57	50	55	84	7	76	57	66	61	97	76	19	57	93	63	45	67	67	46	51	1	77	4	27	30	0	80
40	31	35	15	18	74	94	57	21	5	56	19	71	17	70	30	32	97	96	59	45	69	76	46	50	59	59	28	64	48	81	88	19	41	5	81	12	72	96	65	0

Appendix Figure A6 The amount of flow between facility *i* to facility *k* at period 6 (F_{ik6})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	12	15	9	0	71	5	37	53	97	44	16	80	26	72	99	61	74	42	26	32	90	10	18	11	49	84	3	8	31	53	92	22	30	97	73	10	88	72	42
2	39	0	4	43	94	22	38	87	33	94	95	43	96	11	75	69	81	57	48	62	89	33	80	12	28	89	14	85	65	30	13	96	9	96	35	57	21	31	39	75
3	54	32	0	3	77	55	0	82	10	95	11	27	13	43	84	50	10	71	79	2	71	37	29	18	9	43	28	50	66	56	82	18	66	50	23	1	60	91	28	36
4	23	25	39	0	79	29	5	30	57	53	89	76	19	52	35	59	23	68	25	2	91	49	22	6	33	92	57	70	63	59	60	25	19	82	23	50	24	40	95	69
5	10	91	81	67	0	80	55	27	96	13	33	93	26	34	46	54	73	50	41	17	71	87	28	45	87	52	86	45	18	81	75	67	62	76	25	37	76	75	7	11
6	11	44	93	95	24	0	79	64	91	26	93	84	46	60	28	25	57	40	44	73	5	81	36	17	84	71	98	77	38	44	74	80	97	22	33	4	28	3	41	68
7	91	20	89	78	31	13	0	94	82	9	94	77	49	0	79	5	5	37	79	18	88	73	1	82	66	7	78	2	89	17	21	34	99	7	33	4	95	51	81	40
8	74	45	45	87	30	30	79	0	74	4	34	62	32	66	89	27	29	11	74	76	37	51	38	92	20	13	54	73	49	76	12	14	21	55	11	54	80	49	63	64
9	67	53	19	69	37	87	97	84	0	6	11	39	23	82	69	25	13	41	69	33	25	25	45	56	69	87	76	15	76	58	48	60	55	21	70	63	21	80	42	25
10	3	14	52	62	58	38	49	95	52	0	26	63	16	44	67	50	98	91	57	57	80	72	73	60	7	82	49	24	62	32	47	41	16	34	25	24	80	53	24	55
11	75	87	89	37	3	0	20	77	49	21	0	93	18	38	17	38	65	31	61	98	7	3	60	84	97	90	83	56	74	2	38	51	47	41	97	77	40	49	36	19
12	80	79	12	43	65	69	23	22	33	48	95	0	59	35	71	28	9	37	43	19	43	82	11	47	90	30	79	75	72	94	95	16	86	44	89	76	68	16	55	9
13	21	20	91	93	43	43	81	26	43	81	79	30	0	17	99	72	93	93	7	44	32	52	56	27	7	99	69	36	20	98	60	84	86	71	15	19	49	90	22	45
14	41	64	54	98	88	90	71	36	27	79	85	61	49	0	31	31	77	73	85	53	51	70	76	6	37	82	27	22	19	50	88	2	84	35	42	49	55	55	60	55
15	62	36	50	0	45	5	22	60	64	74	30	99	64	64	0	78	86	49	63	67	32	44	92	22	89	62	23	79	53	86	42	2	58	58	53	36	49	78	95	43
16	15	93	47	90	65	11	19	86	87	77	69	91	93	36	97	0	68	91	58	6	72	69	25	37	76	75	3	77	48	42	50	97	41	78	61	20	16	20	98	56
17	98	93	67	19	32	7	47	5	30	74	8	7	57	98	52	51	0	15	67	95	79	31	11	43	5	73	21	30	49	44	76	84	25	11	21	97	29	99	78	83
18	77	97	11	44	5	17	52	64	88	82	92	51	95	89	12	31	26	0	44	80	72	80	71	41	66	76	63	7	14	79	59	58	32	67	16	7	94	80	45	9
19	16	76	22	71	19	88	1	24	61	35	38	14	12	58	98	74	1	83	0	67	6	48	73	96	87	86	95	74	47	65	57	55	5	44	47	45	31	65	24	4
20	89	71	77	76	64	35	39	15	28	96	11	99	25	48	23	75	21	9	9	0	67	81	51	65	87	32	67	36	1	92	91	49	2	84	40	12	0	27	64	14
21	9	51	14	7	20	45	56	84	57	88	75	42	78	20	94	60	79	82	19	29	0	60	47	37	65	9	48	56	97	15	40	67	85	95	57	48	82	38	14	82
22	48	20	96	30	76	62	28	88	48	51	52	26	33	41	50	40	64	79	16	98	11	0	80	92	29	67	22	91	29	83	82	73	38	55	96	99	83	22	30	37
23	70	20	28	0	39	67	59	62	74	93	21	0	96	40	95	51	24	63	78	26	71	6	0	8	82	11	43	63	78	68	50	31	49	9	29	80	68	28	75	2
24	34	81	86	87	61	58	4	18	38	80	17	17	27	1	71	12	6	32	28	4	16	11	88	0	18	42	3	82	15	45	22	22	63	88	83	30	10	55	83	3
25	87	17	11	57	45	16	9	97	44	93	66	99	92	17	51	23	98	6	92	65	20	34	67	91	0	13	37	80	55	83	21	83	32	72	99	13	86	83	2	47
26	89	37	9	59	57	1	43	16	77	78	36	96	24	21	80	86	21	91	89	22	81	11	86	77	73	0	63	84	19	99	79	77	63	91	89	34	33	29	50	11
27	33	6	73	80	85	83	34	77	9	84	94	16	89	6	38	32	46	20	14	62	0	21	55	54	58	5	0	46	5	85	41	69	54	72	54	27	55	12	34	33
28	40	75	73	8	88	40	87	45	69	35	7	95	14	91	48	35	69	19	64	46	90	39	75	60	87	80	49	0	75	0	88	67	62	90	13	7	55	97	3	3
29	10	13	32	76	52	89	52	96	75	41	74	53	3	40	26	94	54	29	46	97	47	61	25	94	16	32	23	52	0	0	38	75	64	86	33	37	43	99	99	49
30	34	14	0	60	68	46	51	93	20	58	7	1	89	32	38	58	78	56	67	32	11	56	89	92	63	36	62	73	37	0	72	78	57	40	14	45	57	58	41	21
31	90	13	92	60	2	24	21	82	54	5	34	77	44	3	28	40	61	4	17	30	60	1	34	51	44	29	42	78	59	47	0	58	92	35	12	85	23	36	5	52
32	38	99	96	44	76	38	99	96	44	76	80	76	50	10	13	29	51	87	42	14	33	91	83	47	25	9	84	94	61	94	61	0	44	85	61	37	18	63	46	18
33	69	13	36	91	62	83	98	14	40	56	96	46	60	44	82	94	45	66	4	87	40	74	35	57	71	31	13	76	18	54	91	44	0	32	16	50	21	62	36	30
34	12	12	56	74	16	0	16	10	52	53	15	12	67	21	22	99	37	61	85	59	64	45	86	90	30	74	56	42	18	98	33	65	72	0	51	66	91	73	58	16
35	21	68	44	62	24	74	38	48	60	56	5	89	85	76	13	1	75	37	57	18	9	6	65	30	59	6	16	66	86	50	68	41	41	44	0	30	11	36	82	9
36	8	48	27	19	83	5	84	6	75	94	81	82	9	28	91	89	28	95	76	90	83	56	64	40	65	24	74	26	60	19	56	54	83	77	62	0	45	64	47	48
37	37	81	47	68	10	68	15	45	61	11	17	62	69	33	97	6	11	34	35	58	33	50	87	88	5	48	99	89	56	43	12	2	40	70	4	82	0	62	16	24
38	24	60	51	2	82	24	88	92	29	96	38	55	83	97	35	78	30	95	80	15	62	17	21	18	38	42	58	41	86	78	83	37	83	3	84	37	75	0	66	76
39	79	25	99	86	67	16	0	43	31	33	58	60	41	50	90	60	23	90	82	31	49	75	87	75	2	8	9	79	30	86	73	17	71	47	91	66	28	60	0	70
40	32	2	56	35	37	34	30	41	84	35	7	38	85	77	88	82	79	97	9	61	58	71	62	35	26	78	36	61	85	34	4	2	58	20	84	52	74	35	71	0

Appendix Figure A7 The amount of flow between facility *i* to facility *k* at period 7 (F_{ik7})

i\k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	14	20	36	83	21	21	1	67	12	71	74	34	3	13	38	77	12	7	67	82	71	60	72	65	84	19	91	94	20	68	23	9	53	26	34	53	66	61	4
2	32	0	96	38	4	78	31	26	6	7	61	55	87	11	23	75	93	65	87	98	36	57	46	93	72	76	27	81	66	88	1	28	8	5	61	59	90	79	47	91
3	71	97	0	24	89	39	31	42	36	3	41	49	87	97	56	10	18	78	89	51	5	58	21	63	92	85	84	91	32	37	33	23	44	69	30	47	70	17	29	75
4	28	16	26	0	85	8	78	17	36	25	32	84	35	54	25	15	89	3	7	75	41	23	89	68	97	16	78	49	75	86	51	49	26	93	70	11	15	12	74	28
5	56	13	97	37	0	76	73	11	57	84	62	16	93	75	74	22	26	20	53	33	60	57	74	48	28	43	1	58	40	20	89	48	12	17	90	45	5	2	73	29
6	98	74	57	8	50	0	86	48	76	83	68	78	0	31	71	55	32	21	15	82	17	98	56	10	88	38	44	1	14	77	13	67	79	13	3	91	92	92	6	33
7	24	14	46	75	4	76	0	22	47	30	85	92	12	37	73	61	69	42	83	82	65	82	28	37	36	16	73	57	14	92	61	83	2	41	1	46	39	2	51	86
8	53	47	80	16	89	46	88	0	15	46	15	20	1	2	1	17	14	2	96	50	67	37	75	90	26	4	81	34	49	95	57	64	56	67	2	81	28	99	69	28
9	68	13	35	6	73	96	8	16	0	90	8	0	38	49	19	73	16	91	74	22	65	34	47	16	30	79	75	19	75	17	89	38	29	65	26	93	14	31	22	31
10	63	22	18	33	47	35	67	79	6	0	67	8	68	38	8	89	19	13	53	68	15	61	73	66	64	31	34	40	53	75	24	11	61	36	2	65	0	17	81	39
11	72	32	62	34	94	75	13	69	91	17	0	94	43	52	64	52	34	38	70	85	54	42	72	10	23	67	35	36	38	99	6	95	73	13	11	90	16	59	36	9
12	83	59	26	26	9	77	44	59	21	26	13	0	85	70	83	38	50	92	38	40	66	58	92	86	35	51	4	87	77	19	84	52	97	20	35	68	37	79	50	68
13	26	69	38	91	59	82	64	9	11	58	30	22	0	26	67	5	14	69	55	7	44	42	22	11	57	65	27	86	60	64	91	75	79	2	89	22	12	34	83	90
14	85	98	95	88	60	23	64	18	53	67	74	2	11	0	17	65	66	81	9	21	67	52	15	3	27	92	52	54	55	55	92	99	38	85	93	44	50	59	99	46
15	39	44	29	81	62	73	14	18	4	23	32	91	27	83	0	50	89	2	26	23	21	79	42	87	97	65	75	20	37	10	89	80	41	92	34	22	84	15	88	15
16	54	17	54	61	69	10	77	17	29	75	83	74	73	27	84	0	3	64	39	92	98	30	14	72	57	18	61	24	28	97	18	35	32	57	56	70	11	74	45	13
17	97	91	70	35	56	31	32	10	32	33	12	56	9	40	19	43	0	2	5	26	65	56	57	64	95	45	58	63	83	39	55	95	53	19	6	12	37	14	14	47
18	33	45	45	54	21	11	27	4	61	32	30	17	99	70	96	40	48	0	25	2	60	94	20	80	27	56	0	56	45	5	60	80	25	46	34	42	72	41	72	10
19	36	20	91	68	95	96	53	24	96	58	99	13	0	39	56	35	12	90	0	72	77	52	86	78	87	70	34	90	94	22	26	73	96	99	6	31	62	39	99	73
20	25	39	15	83	24	68	39	4	84	92	0	3	69	76	6	3	2	30	13	0	45	20	2	90	8	79	13	48	34	72	97	32	38	92	82	66	19	18	20	43
21	1	87	92	35	47	89	56	88	70	13	82	23	55	63	25	76	18	78	61	83	0	13	43	37	69	87	32	22	37	13	28	45	19	31	83	38	85	56	48	65
22	71	87	51	31	77	35	46	52	14	11	79	97	94	60	44	8	55	57	11	27	99	0	21	81	11	40	63	18	10	81	78	35	27	94	77	16	17	26	40	95
23	59	36	37	69	46	80	29	16	71	9	27	97	40	68	16	98	99	62	79	33	24	2	0	51	49	85	55	91	84	4	41	86	7	77	5	93	15	91	71	37
24	98	85	13	99	67	68	64	0	18	18	80	93	49	86	40	69	91	21	16	30	41	44	94	0	45	61	6	67	41	36	64	79	55	58	24	10	2	15	23	40
25	75	76	62	79	68	77	43	62	30	74	6	83	1	85	55	2	14	90	94	96	51	22	45	88	0	54	84	33	18	48	39	36	26	36	24	11	96	53	40	74
26	79	14	60	97	5	33	16	83	40	26	7	34	11	39	25	27	58	56	16	2	81	93	23	80	23	0	58	18	64	64	11	34	68	97	86	86	80	16	17	78
27	70	68	97	62	27	67	38	97	21	82	62	85	16	57	35	62	27	1	46	10	50	20	7	23	55	5	0	65	25	50	55	45	56	28	63	45	35	62	47	64
28	71	40	27	42	40	56	70	55	83	14	1	96	3	7	11	98	81	4	88	70	85	5	5	85	85	92	65	0	17	72	4	22	66	87	34	36	27	55	36	64
29	55	2	96	11	48	88	88	18	1	24	93	14	63	20	50	65	20	30	63	7	20	63	38	38	15	14	20	36	0	31	41	73	21	6	80	13	83	4	99	33
30	25	35	68	30	64	55	56	85	59	48	57	27	95	40	26	90	53	17	82	7	93	36	69	63	77	91	47	16	53	0	0	94	77	64	51	31	16	85	62	74
31	46	62	84	61	42	55	37	3	27	21	62	71	8	53	42	5	16	26	31	70	80	95	58	56	21	65	71	76	83	53	0	9	28	96	10	22	83	57	84	41
32	11	41	74	42	54	11	41	74	42	54	68	5	25	77	97	8	75	51	86	17	35	54	80	86	89	81	45	34	13	34	13	0	30	47	74	61	25	92	69	66
33	19	19	31	36	4	11	16	26	1	92	37	13	28	84	74	34	85	76	85	23	3	82	54	37	42	81	70	62	4	42	40	0	0	25	46	85	85	50	49	33
34	11	82	53	20	42	52	92	20	32	44	7	96	78	30	22	12	13	5	19	0	47	4	38	17	75	60	59	45	78	89	62	16	0	0	40	36	62	27	28	85
35	39	31	14	27	51	16	5	6	29	79	36	84	88	5	38	61	3	95	34	63	9	46	80	23	29	0	34	59	36	90	59	22	41	50	0	64	73	7	89	5
36	29	14	18	4	70	32	80	49	29	10	22	31	3	2	62	15	46	15	49	74	48	47	92	58	77	33	57	66	73	97	4	35	94	17	80	0	89	35	58	31
37	64	27	1	23	47	51	77	30	69	74	12	51	94	34	15	89	29	30	31	55	32	30	81	83	38	39	10	95	86	20	47	74	76	25	99	95	0	71	74	32
38	78	39	80	63	15	64	71	40	11	9	69	68	57	88	13	58	38	76	17	42	12	23	64	63	60	82	45	81	77	39	62	81	87	21	29	38	68	0	67	63
39	89	85	6	41	71	0	47	62	86	23	53	34	10	11	53	58	9	68	59	23	24	10	33	16	73	80	68	98	58	28	66	31	73	6	72	70	8	76	0	81
40	55	69	48	97	90	25	58	91	53	42	16	96	4	76	81	23	21	29	1	64	37	90	5	63	32	93	56	98	89	71	46	64	95	25	53	36	24	23	14	0

Appendix Figure A8 The amount of flow between facility *i* to facility *k* at period 8 (F_{ik8})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	10	35	32	4	54	15	44	49	37	79	24	21	92	95	97	26	37	5	24	92	2	74	0	45	18	39	78	42	97	74	55	6	69	56	5	77	69	41	78
2	75	0	3	21	89	1	39	10	84	70	98	76	93	16	54	24	63	70	88	16	23	27	88	26	61	7	97	45	76	91	67	98	95	66	75	62	30	72	94	94
3	53	37	0	5	92	56	62	47	47	16	92	72	55	35	64	99	14	17	62	75	9	27	14	80	1	80	58	37	41	72	27	0	83	65	43	94	11	37	26	13
4	80	23	82	0	56	53	31	34	67	19	6	90	40	81	69	2	14	58	44	87	37	46	35	5	79	36	22	33	62	79	25	53	19	54	14	55	20	49	87	35
5	67	79	37	76	0	82	47	72	78	49	87	44	72	49	42	11	47	15	18	15	21	34	37	69	48	78	63	5	7	64	64	11	41	44	91	30	19	53	57	8
6	14	29	64	47	23	0	3	38	48	71	92	52	74	25	5	15	62	77	27	77	80	51	26	25	32	78	88	94	30	26	83	65	7	7	68	35	53	71	12	75
7	81	95	68	69	32	87	0	16	32	47	66	34	10	37	29	80	81	86	12	49	88	82	49	53	78	99	7	67	10	37	59	14	34	98	27	76	60	87	22	46
8	45	10	63	75	68	37	33	0	34	78	33	79	32	26	47	79	75	94	29	29	21	33	92	17	78	16	71	12	22	76	76	25	47	37	80	47	43	18	46	72
9	84	83	69	17	10	25	12	86	0	54	44	18	64	30	40	36	21	28	66	60	79	64	70	79	86	25	54	92	11	8	11	32	49	73	35	19	99	49	23	48
10	37	96	12	0	54	7	40	22	73	0	97	9	40	20	32	49	52	25	85	95	28	96	74	42	8	92	69	67	69	7	87	6	57	61	27	11	64	74	26	87
11	85	10	42	62	11	40	81	52	3	0	0	33	53	45	94	0	78	48	35	4	40	59	53	79	51	17	44	27	62	38	38	67	40	95	35	54	91	74	34	9
12	94	83	46	43	73	89	26	84	36	84	4	0	77	26	67	71	94	46	21	65	69	60	26	91	70	3	19	17	40	26	5	59	33	76	36	30	65	88	19	95
13	35	54	62	52	52	76	63	59	20	89	11	93	0	43	72	4	77	58	74	4	46	1	46	87	76	29	75	72	28	46	59	57	72	13	54	48	68	31	95	81
14	84	66	17	93	88	64	48	61	6	95	16	72	22	0	51	23	1	15	4	93	31	69	20	40	63	36	77	20	93	1	10	18	94	94	79	98	75	57	95	13
15	89	35	47	48	0	65	24	88	44	87	54	66	6	74	0	18	92	99	15	76	14	11	44	95	53	35	9	62	47	26	85	56	36	54	28	97	3	68	53	67
16	6	4	61	80	93	89	81	72	38	36	97	68	77	63	86	0	63	54	99	90	38	60	66	94	13	48	52	64	9	69	7	30	92	46	8	45	28	8	67	57
17	21	38	90	14	73	32	22	83	26	11	59	93	54	44	22	14	0	16	2	20	50	99	51	40	93	85	43	97	54	11	87	31	29	48	90	72	82	93	16	48
18	78	58	72	35	0	42	43	69	20	41	53	76	6	39	38	75	66	0	49	79	38	45	25	39	92	2	36	40	33	15	11	96	2	46	72	56	48	65	7	27
19	13	49	45	44	8	99	62	53	82	16	32	13	23	35	24	94	83	43	0	39	35	55	90	5	81	69	43	47	98	0	38	45	0	80	58	75	62	81	83	20
20	47	30	21	30	3	11	47	50	30	38	60	13	75	3	25	60	0	10	54	0	50	23	66	67	26	0	84	17	98	19	62	89	30	39	71	61	1	60	54	32
21	14	89	53	65	11	73	25	36	63	73	5	86	22	27	12	16	44	87	44	78	0	47	47	4	73	8	56	93	91	29	92	66	91	37	21	68	38	6	19	19
22	32	23	61	58	31	82	62	24	21	69	68	8	42	86	53	49	80	63	59	50	26	0	73	57	80	4	39	14	1	45	28	48	94	46	3	38	0	58	57	22
23	11	1	60	51	18	89	65	4	20	66	45	34	26	25	97	58	18	44	14	1	42	18	0	6	58	31	32	40	59	5	88	77	66	9	61	80	74	24	19	62
24	40	6	6	65	49	79	3	74	50	92	52	94	2	72	0	79	65	77	3	76	9	26	79	0	68	41	20	61	26	4	7	10	21	13	7	40	75	44	22	6
25	45	67	88	46	65	43	32	10	64	42	85	21	68	31	3	72	71	98	49	52	40	68	17	64	0	25	33	31	21	16	50	54	17	1	23	99	45	13	80	33
26	22	89	35	25	34	87	75	80	10	61	55	77	52	59	34	29	43	46	60	4	8	4	19	59	26	0	41	78	4	62	20	82	29	49	95	42	45	31	61	58
27	40	98	74	0	35	30	40	70	55	13	10	15	55	80	73	86	14	59	18	53	52	71	53	51	74	18	0	93	9	44	87	61	51	76	73	8	80	72	4	43
28	98	73	61	28	7	50	58	68	49	47	78	26	95	14	52	32	10	46	63	35	27	77	42	72	81	18	15	0	95	38	33	55	76	95	61	44	81	76	79	79
29	54	50	30	27	59	12	35	99	67	1	95	40	73	32	20	11	61	15	26	19	74	82	56	61	47	15	82	73	0	57	7	29	89	21	64	86	54	32	5	32
30	11	5	0	92	35	45	37	74	54	83	66	45	80	49	3	63	58	56	87	64	65	5	4	66	2	56	37	59	11	0	27	71	29	95	86	9	13	28	48	61
31	98	87	52	31	97	74	54	33	42	30	94	84	22	27	37	87	52	71	45	28	60	94	51	35	19	37	86	27	5	7	0	56	6	83	98	7	16	27	2	82
32	92	9	93	80	68	15	30	87	90	99	30	89	72	54	74	23	83	53	39	18	17	14	29	5	47	87	99	41	61	85	51	0	78	70	76	30	17	33	38	22
33	33	73	54	59	42	43	7	59	87	20	92	42	33	38	65	61	14	34	54	95	75	61	41	71	38	79	65	82	0	35	64	69	0	61	54	73	65	68	75	81
34	2	27	80	6	19	28	64	80	80	38	21	8	43	29	13	34	80	23	19	79	60	52	49	82	27	33	16	70	38	86	11	69	5	0	31	80	21	56	38	76
35	50	45	98	62	70	66	66	41	79	81	31	12	59	78	92	50	26	66	92	19	58	50	32	10	45	94	22	70	88	55	58	4	9	13	0	76	11	81	86	28
36	20	26	50	95	37	76	99	27	18	11	3	18	42	63	79	56	60	60	42	77	60	32	81	47	5	21	90	56	8	1	19	37	83	15	43	0	23	35	25	6
37	92	12	22	2	67	98	19	24	3	96	27	75	7	63	87	73	56	78	2	31	82	85	93	69	55	5	65	65	74	97	80	79	76	28	17	63	0	10	94	24
38	31	11	83	2	57	45	31	60	49	55	74	91	95	45	31	83	17	92	77	27	3	81	21	79	84	46	41	25	73	14	76	61	59	13	45	97	83	0	44	91
39	23	75	11	85	36	68	88	65	15	16	11	81	35	73	36	85	59	9	75	90	46	58	29	21	66	31	31	33	18	82	97	76	64	77	45	35	97	96	0	32
40	22	63	19	33	67	83	24	80	31	0	54	78	40	1	80	82	90	76	23	36	78	30	89	53	82	3	54	19	4	92	13	93	70	84	32	66	10	66	78	0

Appendix Figure A9 The amount of distance between location *j* to location *l* at period 1 (D_{jl1})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	51	11	55	54	69	14	17	63	20	45	89	74	93	62	21	82	50	62	87	38	12	83	69	87	86	66	12	15	97	41	71	30	86	48	26	71	2	91	89
2	31	0	93	59	46	95	53	22	32	59	36	77	89	58	72	98	1	10	72	17	63	9	52	1	21	81	40	76	66	44	91	81	23	6	90	52	5	88	22	41
3	41	2	0	14	88	75	59	36	8	19	68	99	73	50	40	22	33	24	18	88	35	12	37	49	85	44	10	41	85	4	52	39	8	97	44	97	94	6	24	62
4	22	84	20	0	47	75	98	52	59	77	85	58	62	16	79	88	33	86	17	84	85	40	31	30	73	23	71	11	27	75	38	59	0	68	94	29	90	71	56	9
5	47	66	33	94	0	70	65	97	39	61	58	48	3	47	75	75	85	40	76	43	23	34	18	56	41	65	65	13	50	30	8	34	60	73	12	57	18	53	6	29
6	4	92	60	74	54	0	21	76	39	9	86	42	59	64	33	34	57	71	30	34	50	1	17	39	60	44	87	75	3	9	30	21	33	22	31	44	44	37	20	29
7	53	46	43	63	17	31	0	47	11	16	46	6	32	8	3	40	15	68	8	24	84	89	84	4	48	30	84	70	71	28	92	64	50	69	28	66	95	21	58	17
8	5	15	24	44	14	39	76	0	78	20	93	81	47	82	70	27	47	6	4	64	65	99	55	28	59	2	65	48	45	62	53	99	47	12	65	59	3	59	84	76
9	33	99	28	95	32	74	98	23	0	26	17	87	68	69	22	80	36	45	76	44	19	7	50	70	3	37	50	88	57	14	38	95	87	37	69	32	97	12	30	18
10	46	47	65	75	60	82	18	58	71	0	67	45	38	78	29	49	24	67	24	70	30	46	8	1	61	3	63	91	39	0	91	79	2	77	72	28	64	70	70	67
11	98	22	85	73	55	55	85	36	12	84	0	54	36	10	32	49	6	24	90	72	35	93	6	11	89	87	46	49	34	50	17	79	67	56	22	75	62	72	14	43
12	87	36	60	60	69	50	18	57	74	70	55	0	88	12	50	48	88	62	40	16	47	19	0	67	10	42	13	77	24	52	3	69	70	31	19	4	45	67	54	75
13	50	76	84	79	69	12	24	57	58	83	18	82	0	88	4	76	95	88	69	16	73	96	79	64	53	54	91	16	20	33	28	55	66	0	5	11	29	83	9	97
14	84	61	83	74	71	38	6	70	0	49	55	2	5	0	76	2	39	3	78	30	6	64	73	60	55	20	57	91	87	50	13	39	36	0	98	21	12	21	59	10
15	69	66	21	7	21	25	36	18	95	76	40	75	15	91	0	66	59	60	56	57	21	58	80	69	13	22	12	6	1	36	98	16	32	82	48	41	94	35	58	98
16	78	13	15	93	99	8	90	47	31	87	48	9	41	6	1	0	99	4	67	72	8	73	48	39	47	79	69	98	82	69	57	9	50	91	94	78	44	92	29	40
17	65	25	71	87	2	68	23	20	58	69	3	76	68	39	21	40	0	11	58	96	13	37	99	34	48	82	35	44	10	28	32	68	59	30	73	68	52	15	11	98
18	16	71	13	32	84	54	80	11	12	49	81	55	24	46	27	29	55	0	32	31	98	97	32	79	15	72	12	51	36	72	27	44	56	39	16	68	38	33	50	25
19	25	12	88	77	15	56	86	91	7	58	66	47	71	87	94	10	93	79	0	2	13	23	88	37	8	74	76	68	57	35	20	85	48	31	9	45	29	59	12	29
20	71	99	14	97	97	14	88	77	41	89	39	68	92	99	72	94	66	33	58	0	1	88	9	17	83	26	47	41	14	60	14	44	18	97	27	50	23	7	94	38
21	34	55	74	48	63	53	62	76	9	6	49	35	9	62	4	39	93	31	28	0	0	45	67	12	56	55	43	13	95	77	68	98	41	7	57	52	62	3	28	41
22	87	35	86	16	73	59	94	15	20	25	78	31	65	26	20	68	28	2	52	51	40	0	7	37	85	71	12	12	64	30	18	57	55	36	19	67	71	92	76	44
23	90	64	51	97	51	67	32	1	70	35	13	46	77	51	26	46	59	2	82	20	70	41	0	18	90	82	64	44	82	37	27	36	21	7	67	69	32	94	79	19
24	71	31	4	74	70	50	71	55	80	81	83	72	15	47	31	69	55	99	6	47	76	26	30	0	25	54	84	87	32	4	58	48	43	84	21	14	43	40	39	26
25	57	2	79	80	48	30	11	35	3	79	37	97	97	2	69	71	83	47	60	89	75	62	76	70	0	39	70	52	89	8	30	16	94	87	28	83	38	6	95	35
26	57	1	23	1	63	27	89	1	36	92	66	62	48	88	36	94	13	85	86	64	63	42	14	29	86	0	52	46	0	70	90	27	89	89	46	91	78	70	60	97
27	86	83	42	98	97	40	39	4	82	27	69	68	99	58	38	76	91	32	11	84	2	73	39	94	23	77	0	63	5	3	80	20	32	26	24	99	81	33	36	99
28	69	78	35	78	32	81	31	67	35	6	76	2	21	77	81	75	93	77	88	61	74	45	2	47	49	95	60	0	93	44	28	91	37	47	83	77	37	10	61	75
29	42	35	93	56	7	74	96	12	61	95	39	37	70	74	31	40	56	89	52	1	69	38	93	53	34	71	43	74	0	84	37	9	78	85	64	66	5	42	31	66
30	77	22	95	27	65	28	22	4	67	8	16	9	72	61	88	23	43	41	72	33	97	77	57	63	45	58	61	39	20	0	78	80	42	70	73	3	78	28	90	83
31	25	32	75	92	53	37	79	84	63	17	85	35	55	58	59	55	19	79	24	46	12	72	99	67	1	46	87	7	54	82	0	74	42	68	82	35	47	99	69	39
32	11	36	38	92	94	38	71	92	67	67	5	65	1	81	74	24	88	29	25	23	56	64	61	69	90	2	95	87	59	54	2	0	20	13	11	98	2	99	8	13
33	43	32	77	9	92	45	50	58	32	44	81	25	49	41	92	84	53	86	18	9	61	92	87	69	43	51	72	30	13	69	83	64	0	38	78	91	16	27	0	46
34	9	12	70	73	31	46	75	73	76	29	22	29	91	79	79	45	95	93	74	40	61	9	70	68	26	25	41	83	86	77	52	73	16	0	80	7	55	19	8	76
35	80	31	62	91	16	15	82	13	13	48	36	3	67	99	57	3	23	60	0	12	12	99	37	65	44	3	96	11	61	45	75	0	44	41	0	58	43	73	35	18
36	4	31	87	4	30	77	89	53	65	62	18	63	72	96	39	22	53	63	92	19	42	36	5	45	84	7	81	16	89	62	25	67	93	89	85	0	11	53	75	53
37	37	63	61	74	33	52	37	25	76	56	2	83	70	43	21	69	51	34	99	99	67	78	58	30	93	29	42	97	77	82	48	54	48	96	78	74	0	54	67	37
38	66	77	44	92	12	21	60	51	12	19	18	61	62	55	31	3	79	0	65	0	62	56	65	6	33	35	60	25	54	76	54	64	86	3	34	61	88	0	80	4
39	63	28	0	72	29	99	78	12	10	10	65	3	60	36	84	83	58	67	30	20	34	4	33	27	19	48	50	57	34	54	99	83	7	68	54	44	80	58	0	6
40	24	50	15	5	30	44	10	45	99	24	55	89	39	55	31	77	36	99	98	9	23	58	80	84	38	43	42	52	62	74	41	15	39	15	91	75	71	28	8	0

Appendix Figure A10 The amount of distance between location *j* to location *l* at period 2 (D_{jl2})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	64	47	83	59	59	12	20	58	57	71	73	99	69	96	54	51	14	72	58	12	4	94	97	63	81	59	56	18	0	17	45	30	13	75	72	46	63	66	6
2	90	0	34	97	28	42	8	87	10	63	93	71	21	5	59	48	13	47	95	47	44	23	89	65	78	89	12	88	44	46	53	72	91	28	57	75	62	34	73	52
3	53	71	0	20	44	3	39	64	3	0	58	50	70	7	81	75	48	6	83	50	95	69	9	22	11	67	54	50	35	72	10	43	9	79	81	1	81	25	50	68
4	67	68	63	0	98	70	7	27	62	23	92	45	8	99	63	72	34	66	25	63	28	55	66	45	66	98	36	82	18	62	62	5	37	31	63	2	54	48	76	61
5	86	7	30	48	0	67	35	52	5	23	14	89	54	31	92	88	98	11	3	68	62	52	86	22	53	58	11	23	46	32	85	93	39	70	18	69	71	62	37	86
6	4	1	73	28	24	0	51	23	62	24	47	90	86	61	29	35	13	77	24	96	10	77	65	30	23	93	53	90	42	50	90	22	64	53	98	0	16	12	20	88
7	81	82	68	79	39	66	0	10	68	20	0	25	74	25	10	0	66	65	54	28	83	87	71	1	81	17	74	56	77	66	24	6	7	27	38	46	30	47	28	43
8	85	42	53	86	8	17	11	0	2	25	90	45	2	99	53	50	1	12	68	96	35	86	96	11	23	89	43	50	60	94	61	27	80	59	31	97	8	40	32	58
9	98	60	86	30	40	84	17	84	0	86	10	58	65	24	64	59	67	92	13	23	64	44	5	40	80	73	76	57	96	93	70	21	7	96	83	6	10	92	45	19
10	90	56	80	62	42	10	2	82	26	0	48	68	1	12	71	58	47	51	26	89	10	53	44	37	96	86	97	88	0	65	18	70	47	77	1	20	0	33	84	75
11	80	22	36	88	60	74	4	67	85	90	0	2	70	78	72	60	88	22	7	92	14	6	8	5	13	73	31	84	24	91	74	76	81	75	92	50	86	76	46	47
12	33	1	14	23	85	7	70	93	65	92	64	0	30	40	44	60	59	77	8	72	53	64	89	35	46	1	97	0	43	73	60	96	47	59	15	0	20	58	22	84
13	39	99	33	82	63	30	1	83	85	81	24	51	0	92	67	26	17	87	37	52	92	94	98	33	6	8	9	35	44	64	55	34	19	98	43	86	80	26	30	26
14	14	99	79	93	79	97	25	37	80	5	75	6	62	0	79	77	58	88	31	60	59	15	67	32	43	90	42	6	11	35	64	39	32	45	68	34	14	20	59	27
15	6	50	72	90	96	88	9	61	2	27	79	94	11	92	0	75	46	88	27	95	37	4	81	28	73	12	44	76	52	90	95	53	23	35	5	95	86	9	64	39
16	0	29	2	82	74	82	20	61	86	65	0	21	60	41	29	0	48	75	40	96	18	19	95	46	27	80	22	13	11	48	84	74	92	10	49	68	7	5	69	71
17	22	12	60	1	84	86	19	43	4	82	84	3	28	63	76	51	0	93	33	32	58	34	40	70	63	50	94	77	96	28	48	57	7	70	39	24	11	38	37	35
18	48	82	78	35	42	14	49	32	23	69	83	38	53	35	48	22	76	0	98	59	81	77	4	61	13	41	1	55	26	26	24	26	51	90	8	37	56	21	98	6
19	96	78	58	10	82	89	91	45	11	2	88	82	66	97	3	62	7	4	0	6	41	39	48	53	81	94	25	50	44	66	58	55	30	57	26	41	65	47	53	25
20	80	21	94	80	9	15	98	68	59	59	72	77	83	34	35	70	88	98	76	0	5	53	3	76	36	15	15	24	74	52	42	1	38	31	59	96	87	53	38	76
21	21	84	37	84	60	62	49	48	85	6	49	21	82	84	24	28	78	47	62	73	0	83	74	72	85	49	45	66	61	67	68	35	57	59	27	79	3	61	83	21
22	52	14	88	84	10	84	68	26	33	6	9	47	27	2	39	32	58	37	27	15	5	0	80	59	18	41	50	63	43	66	8	69	71	89	63	0	99	32	16	13
23	54	70	1	14	73	81	76	60	57	19	46	2	9	85	49	39	16	84	6	30	74	16	0	5	4	55	67	34	11	26	45	57	85	0	72	20	19	50	62	67
24	44	70	77	29	94	89	63	40	9	88	18	68	40	59	94	72	97	2	56	33	23	66	70	0	32	87	52	2	12	45	43	31	89	8	82	70	81	91	87	89
25	98	86	85	72	15	49	61	48	30	65	25	37	85	63	51	26	58	31	53	73	80	84	92	38	0	67	68	94	65	2	77	97	3	88	92	89	37	83	37	99
26	73	2	48	94	5	11	30	65	88	39	38	54	0	18	93	52	10	14	2	53	21	16	28	87	95	0	73	77	26	24	73	87	47	92	76	46	46	35	3	13
27	7	43	38	12	0	0	68	24	19	42	76	64	36	62	96	21	84	11	34	92	79	67	2	74	3	21	0	15	84	24	90	18	74	53	17	78	79	69	2	18
28	80	54	3	13	19	81	89	36	60	63	37	66	53	90	95	61	73	84	22	67	83	73	80	15	42	62	74	0	83	43	42	28	9	16	29	18	2	47	10	95
29	90	4	4	68	95	19	97	40	65	80	77	90	52	26	92	5	6	15	71	4	22	15	0	88	53	17	73	88	0	45	16	20	54	57	14	17	12	73	96	67
30	93	69	54	18	53	97	4	95	4	81	27	16	12	12	3	68	67	67	91	18	52	54	96	44	78	5	10	89	53	0	53	17	81	79	68	88	22	38	65	17
31	9	93	35	98	60	99	10	72	19	44	65	46	15	65	38	74	85	58	74	65	10	53	27	4	43	31	51	4	5	61	0	88	76	49	92	89	98	55	70	25
32	26	63	79	68	1	78	60	37	9	71	57	56	0	8	96	29	95	90	61	32	44	41	92	4	10	65	9	30	90	94	72	0	93	69	17	44	89	42	55	32
33	96	93	80	85	41	86	59	27	63	29	21	11	21	15	61	32	47	1	67	5	5	79	90	26	24	75	99	61	22	82	11	75	0	63	62	71	89	24	58	51
34	72	35	40	98	87	8	23	3	65	80	23	68	65	59	56	25	15	66	49	52	53	86	76	75	7	92	24	70	3	57	93	31	0	0	18	21	7	89	26	61
35	5	26	55	44	91	43	13	66	36	31	46	30	83	75	64	27	10	59	26	43	57	82	28	4	23	98	74	63	57	91	25	12	17	75	0	63	3	86	82	6
36	41	9	69	30	23	53	32	72	25	43	31	51	47	38	98	24	69	21	86	24	81	39	70	57	11	9	54	52	61	68	61	87	76	74	93	0	95	62	63	66
37	93	84	92	47	64	18	98	98	44	58	43	10	57	91	85	29	94	65	41	85	31	61	56	99	5	78	34	89	47	26	20	97	56	11	68	37	0	8	70	17
38	11	82	34	88	58	49	26	61	73	89	4	84	7	98	38	40	10	10	60	16	79	46	38	58	20	16	70	71	8	79	70	89	1	26	12	82	33	0	28	29
39	45	35	56	24	43	77	55	66	94	23	20	66	67	29	12	86	46	86	16	48	63	74	67	72	68	25	32	12	71	91	49	1	27	41	24	31	63	66	0	65
40	93	74	48	48	53	10	41	62	46	42	56	42	71	94	75	61	35	30	58	92	91	61	45	64	65	0	78	66	6	74	16	21	33	94	27	27	20	35	28	0

Appendix Figure A11 The amount of distance between location *j* to location *l* at period 3 (D_{jl3})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	72	12	39	86	51	64	30	85	20	69	74	86	27	33	19	18	48	80	13	28	34	0	52	57	62	59	19	13	20	73	62	22	20	85	97	53	43	2	41
2	33	0	65	9	12	85	36	93	66	27	15	63	23	94	46	34	45	4	64	20	36	38	49	17	84	21	11	48	12	81	41	84	74	90	88	57	70	87	68	40
3	41	65	0	66	85	84	55	70	71	5	68	14	54	95	25	35	3	58	48	38	12	6	58	16	89	20	62	28	92	81	54	64	49	18	95	43	49	97	76	34
4	77	9	5	0	0	13	85	88	60	92	36	89	55	74	46	26	9	46	24	6	21	28	63	42	94	84	52	47	43	0	19	23	25	39	94	33	99	81	15	53
5	45	27	11	53	0	81	81	7	37	80	62	33	19	5	56	99	56	25	86	15	34	22	94	52	47	15	67	8	45	61	5	13	67	2	66	69	50	40	94	11
6	2	42	77	81	51	0	70	97	46	61	83	57	0	40	88	30	4	98	26	92	35	18	78	87	3	8	99	82	60	14	61	79	25	79	3	88	11	76	85	52
7	28	81	32	22	88	6	0	29	29	76	74	6	55	3	99	90	85	48	15	66	41	84	86	36	5	83	73	12	2	36	15	11	38	76	78	55	3	6	52	80
8	48	91	29	28	57	22	20	0	80	42	39	91	98	32	96	43	20	72	27	23	71	33	40	20	0	51	14	72	75	87	71	83	82	45	54	95	29	54	6	17
9	3	29	36	10	35	88	70	70	0	20	49	3	85	88	83	24	87	44	98	85	30	10	22	71	9	41	42	87	91	77	92	33	6	87	3	81	0	33	3	63
10	70	44	70	9	69	26	4	34	70	0	89	38	49	21	30	57	6	53	9	3	54	84	7	16	36	82	24	52	14	48	31	50	97	32	25	60	73	17	86	23
11	81	24	56	60	75	75	2	42	18	58	0	73	69	0	58	66	16	2	27	42	89	90	83	95	91	57	77	74	8	22	59	63	91	59	19	32	3	3	21	3
12	38	80	52	84	17	9	25	82	76	45	38	0	82	41	41	7	27	17	74	52	11	48	78	0	6	72	26	75	33	9	22	32	6	83	37	92	70	33	58	5
13	20	31	55	38	79	37	64	58	93	38	6	76	0	12	37	84	58	36	97	67	75	55	9	84	35	2	71	31	80	44	38	82	14	30	90	16	0	83	57	53
14	78	20	14	41	38	57	91	73	37	47	49	29	72	0	83	46	49	25	36	66	92	28	45	29	56	99	40	58	39	71	46	86	38	74	98	46	80	22	17	71
15	46	93	34	38	37	85	13	35	1	26	2	91	19	59	0	92	21	78	73	47	13	24	35	4	60	18	1	25	36	40	94	41	20	4	65	60	4	54	79	35
16	52	6	60	63	28	37	40	91	64	1	58	30	8	79	28	0	27	69	7	97	79	7	17	3	15	72	19	83	30	65	44	93	36	65	31	56	1	61	77	79
17	46	98	40	90	78	43	37	77	7	71	45	31	74	19	20	87	0	23	74	83	62	36	13	12	60	69	99	25	14	32	45	22	44	35	22	96	6	18	42	23
18	70	20	16	40	26	25	14	19	66	68	14	34	95	1	56	65	11	0	72	49	72	97	99	56	7	89	16	47	34	91	49	43	22	26	47	29	36	29	51	45
19	16	41	4	31	15	2	23	95	55	24	42	24	8	6	72	92	3	3	0	0	55	52	63	61	97	91	19	48	29	26	84	68	40	99	27	0	90	48	77	28
20	94	31	65	17	7	34	11	30	29	45	86	64	2	48	12	47	96	90	62	0	98	29	88	77	37	78	45	23	15	25	77	46	99	86	54	43	59	36	4	67
21	85	17	48	73	53	85	30	30	3	81	87	14	36	87	18	39	70	15	76	25	0	71	44	39	8	37	1	41	16	28	57	55	65	43	18	34	4	83	27	10
22	79	33	69	71	11	41	94	18	99	27	17	64	50	89	39	82	41	60	41	94	70	0	95	21	6	66	97	10	12	91	86	99	63	36	79	6	78	76	10	48
23	31	12	3	42	79	94	99	32	5	91	62	1	40	49	2	1	32	73	1	19	68	79	0	3	17	76	17	46	22	57	37	85	80	75	67	55	96	9	90	35
24	36	98	5	59	29	75	78	72	11	49	21	69	34	54	71	48	85	17	12	3	11	53	80	0	66	28	37	65	66	25	76	28	47	84	49	25	52	66	64	46
25	82	91	44	37	2	49	77	41	24	55	24	97	92	41	68	30	59	96	43	82	65	54	69	17	0	77	7	42	12	79	75	19	38	70	64	40	72	51	14	53
26	94	3	36	7	13	93	54	12	63	37	64	43	22	87	63	90	90	61	55	28	52	83	39	99	39	0	8	98	70	71	77	49	93	91	9	90	75	55	20	15
27	45	95	45	46	32	31	15	24	27	78	97	45	62	36	66	52	41	33	76	2	12	57	68	25	13	82	0	27	42	71	2	74	63	38	74	54	70	12	71	51
28	93	10	75	4	83	91	62	54	36	36	8	6	75	53	69	45	77	20	27	80	13	22	40	6	56	57	76	0	67	94	35	12	89	20	5	19	54	98	20	84
29	31	26	59	35	57	19	87	57	26	28	68	60	28	97	67	88	11	6	48	79	99	56	30	80	61	31	86	83	0	24	84	55	40	2	0	95	41	10	75	50
30	32	94	93	46	90	91	97	2	30	44	55	87	54	94	0	72	8	59	61	61	47	67	34	18	67	81	29	44	31	0	21	80	91	17	26	32	3	24	60	73
31	58	52	94	62	71	71	34	61	11	4	78	93	13	51	52	23	82	23	47	98	56	43	7	52	71	12	15	70	29	21	0	86	82	87	80	1	63	91	64	81
32	49	17	63	84	8	33	10	62	61	60	62	5	60	83	80	45	25	4	4	34	22	2	27	83	0	1	40	12	79	40	22	0	61	82	26	61	37	33	92	22
33	61	42	42	43	69	99	44	98	61	30	49	58	43	80	23	88	86	61	44	30	65	13	8	13	99	15	60	26	1	57	13	31	0	22	78	2	39	7	71	43
34	61	14	32	62	31	58	68	41	57	89	10	43	46	20	99	68	73	60	88	45	4	1	24	49	96	23	93	25	29	86	16	19	15	0	77	3	92	70	9	53
35	81	33	21	87	67	79	99	26	86	83	83	81	3	98	62	70	68	22	81	75	82	54	28	9	66	15	41	31	37	14	6	25	95	82	0	21	98	76	23	45
36	49	48	15	33	26	42	67	49	75	3	48	11	11	80	92	8	4	11	36	18	87	75	16	34	82	29	89	57	72	19	3	7	77	23	46	0	57	39	15	2
37	72	38	60	46	11	98	24	46	35	48	2	98	28	35	91	19	62	73	70	43	21	20	86	26	15	67	36	83	66	73	15	96	2	62	94	27	0	12	31	68
38	12	43	2	56	40	6	90	61	45	14	28	78	32	49	82	32	42	58	85	36	99	18	18	73	76	90	30	65	8	44	77	15	35	62	86	49	76	0	5	41
39	11	40	64	8	51	81	76	16	95	73	94	86	81	22	19	33	98	56	14	92	23	83	59	68	53	79	94	17	6	9	40	52	9	45	1	14	52	62	0	55
40	1	50	90	24	43	56	60	50	97	74	71	96	7	7	55	62	95	85	9	1	91	9	15	15	95	61	95	42	77	0	99	13	99	43	74	31	13	56	76	0

Appendix Figure A12 The amount of distance between location *j* to location *l* at period 4 (D_{jl4})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	41	95	73	20	7	48	92	98	86	16	91	86	92	59	19	66	0	18	79	35	71	31	50	31	19	4	99	37	26	87	59	82	23	58	54	56	15	14	87
2	73	0	32	54	11	7	59	44	85	75	55	44	15	20	30	51	43	62	59	97	93	17	66	57	83	5	52	28	21	5	36	2	54	13	62	59	79	5	58	55
3	44	51	0	7	70	84	25	43	39	25	11	21	1	69	90	29	94	82	5	67	20	75	26	44	25	10	23	76	26	30	10	16	45	33	15	31	49	8	46	44
4	42	2	24	0	9	29	17	98	42	98	37	6	23	60	62	87	74	10	90	97	47	77	3	15	6	60	11	8	47	57	54	32	58	5	58	53	23	18	47	64
5	17	21	26	55	0	0	16	80	75	95	0	32	71	73	27	60	50	34	65	8	87	87	40	73	69	19	72	44	38	42	65	30	96	24	93	45	29	77	95	49
6	61	44	33	59	7	0	15	39	33	64	47	22	29	89	28	1	62	52	55	96	84	37	18	4	66	35	63	12	56	31	87	96	63	66	68	99	71	76	15	87
7	96	31	97	89	71	88	0	19	74	82	45	29	72	5	1	15	72	76	52	48	12	16	49	38	79	5	54	58	60	56	17	11	83	13	26	18	32	83	42	44
8	27	72	52	29	31	40	76	0	9	30	56	17	30	30	22	64	30	37	61	53	60	77	64	35	31	9	81	71	66	80	54	49	74	44	86	74	55	79	41	44
9	90	51	79	75	44	17	55	67	0	14	21	28	87	75	44	62	47	62	53	75	42	97	36	8	55	51	0	90	45	53	65	93	60	37	28	7	12	11	15	61
10	10	58	10	29	92	83	46	4	63	0	71	56	82	84	12	11	60	12	32	84	8	72	0	99	64	90	49	56	93	91	8	13	4	33	56	42	98	42	12	74
11	46	42	10	14	95	30	18	77	16	86	0	75	16	14	90	36	48	10	30	72	48	77	46	68	69	5	28	83	79	50	42	48	72	6	65	74	32	67	5	53
12	32	84	19	80	90	15	35	49	30	66	84	0	98	79	39	48	80	40	90	67	19	63	63	8	55	60	63	91	7	89	5	20	94	27	89	32	78	61	68	93
13	94	73	39	97	64	67	57	56	72	78	12	50	0	91	65	38	87	31	51	72	61	52	60	37	51	88	30	69	70	6	60	24	26	49	27	39	22	94	54	59
14	66	88	22	19	81	64	11	62	40	93	90	69	51	0	73	67	77	8	30	82	18	26	29	64	7	77	85	81	74	66	27	8	67	92	75	97	47	23	74	71
15	57	41	25	62	57	92	44	31	66	73	92	68	56	37	0	65	5	90	37	76	80	73	14	8	37	16	24	23	84	95	72	97	29	26	93	30	68	58	53	8
16	72	47	44	56	21	75	84	51	76	1	76	19	15	65	55	0	38	47	23	89	91	66	86	87	93	7	40	78	41	52	76	91	32	66	49	68	60	56	92	10
17	63	13	4	71	92	52	15	4	70	3	28	67	18	41	58	78	0	62	52	2	61	88	58	68	74	4	30	92	55	54	72	77	48	98	7	61	46	93	17	70
18	96	24	17	26	65	16	63	95	39	65	90	6	14	66	80	17	34	0	66	34	15	82	81	6	40	96	74	39	86	89	28	55	36	21	95	29	95	96	26	32
19	51	4	8	76	94	4	88	72	33	10	29	82	22	93	89	39	32	58	0	7	39	8	4	25	39	79	25	43	59	54	89	93	37	71	50	49	48	78	37	57
20	76	1	9	95	26	46	16	92	75	44	73	58	80	55	78	70	10	65	37	0	31	14	13	45	53	26	29	65	28	94	82	0	53	47	39	45	44	68	36	7
21	0	20	46	59	53	56	45	12	11	59	16	91	9	26	66	84	5	84	63	3	0	90	50	21	43	19	74	28	62	27	76	92	66	93	2	91	99	81	64	32
22	94	93	27	7	84	35	83	90	67	36	10	67	10	97	55	62	79	69	41	22	80	0	55	11	26	97	73	41	13	63	74	88	9	69	47	63	71	56	37	38
23	92	86	8	62	32	75	96	11	94	31	31	2	48	53	28	86	99	73	94	94	27	12	0	33	42	52	87	9	30	41	34	13	16	97	58	98	93	26	17	18
24	46	95	44	19	81	24	58	15	88	18	44	48	18	4	83	46	9	71	80	77	15	13	84	0	26	84	70	96	96	92	69	95	19	93	18	77	53	49	49	79
25	36	57	86	35	52	96	80	10	79	48	29	36	74	70	59	52	33	87	95	44	2	84	30	84	0	10	7	63	91	30	79	46	68	24	80	83	68	34	83	46
26	15	40	34	10	61	98	55	84	7	0	40	41	53	4	83	59	35	92	71	60	6	59	32	92	76	0	7	30	80	67	27	24	58	97	61	15	84	79	56	71
27	89	24	17	79	51	78	97	88	88	25	14	61	4	56	51	68	17	44	68	2	21	18	50	98	29	77	0	78	16	81	24	41	22	90	11	23	25	6	54	40
28	51	47	89	94	65	59	66	2	97	50	60	9	46	45	34	55	78	77	98	28	20	71	91	86	11	67	74	0	92	10	15	21	43	95	31	22	99	87	6	65
29	1	94	53	75	18	31	71	10	60	42	64	13	26	39	82	55	56	10	86	31	90	12	13	40	72	38	43	54	0	81	83	38	50	43	32	44	40	31	3	12
30	85	28	30	76	16	48	30	55	90	92	81	27	29	97	69	75	59	88	63	87	73	5	66	49	66	38	85	97	20	0	16	78	96	7	26	44	52	14	66	71
31	94	34	19	43	33	28	21	96	95	33	36	4	65	46	1	93	33	18	30	81	67	8	26	52	20	11	27	96	60	74	0	30	87	6	95	55	57	14	51	68
32	69	61	40	97	56	43	84	40	11	94	46	9	70	50	96	8	87	47	54	36	55	18	21	64	17	7	5	52	51	24	23	0	22	53	79	9	15	10	30	58
33	30	70	35	20	89	93	3	21	33	29	61	56	88	23	72	52	15	10	11	13	71	34	18	98	5	42	88	88	85	62	80	29	0	15	69	20	57	54	97	40
34	42	17	31	71	3	80	26	17	71	2	83	1	85	54	73	1	20	34	80	96	83	99	54	38	82	22	51	48	80	87	52	57	26	0	97	29	22	56	95	81
35	31	61	95	28	33	1	60	69	32	90	69	25	55	16	15	36	69	13	97	92	92	95	64	85	28	64	92	33	28	9	89	88	98	75	0	83	63	53	4	99
36	96	16	86	90	17	40	48	74	90	41	36	89	29	29	7	57	4	59	31	67	44	60	61	30	17	53	49	29	48	91	80	40	45	85	53	0	26	36	4	82
37	84	53	62	84	35	55	54	33	45	90	43	33	57	24	3	79	24	45	82	79	6	4	19	92	74	10	83	31	17	86	2	84	38	25	84	45	0	59	24	92
38	31	80	85	24	93	80	73	10	79	53	87	92	2	88	37	27	1	45	88	20	60	58	76	99	80	34	87	97	6	71	90	42	56	63	73	52	26	0	61	24
39	31	81	31	78	91	39	17	44	22	1	83	17	88	77	1	23	24	73	30	15	13	88	63	18	64	67	87	34	1	1	50	5	54	12	59	61	76	0	0	9
40	59	94	32	13	33	11	78	90	8	95	48	71	73	44	90	24	64	78	56	41	27	65	34	94	45	57	16	91	22	30	41	48	20	24	39	31	18	22	66	0

Appendix Figure A13 The amount of distance between location *j* to location *l* at period 5 (D_{jl5})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	51	11	55	54	69	14	17	63	20	45	89	74	93	62	21	82	50	62	87	38	12	83	69	87	86	66	12	15	97	41	71	30	86	48	26	71	2	91	89
2	31	0	93	59	46	95	53	22	32	59	36	77	89	58	72	98	1	10	72	17	63	9	52	1	21	81	40	76	66	44	91	81	23	6	90	52	5	88	22	41
3	41	2	0	14	88	75	59	36	8	19	68	99	73	50	40	22	33	24	18	88	35	12	37	49	85	44	10	41	85	4	52	39	8	97	44	97	94	6	24	62
4	22	84	20	0	47	75	98	52	59	77	85	58	62	16	79	88	33	86	17	84	85	40	31	30	73	23	71	11	27	75	38	59	0	68	94	29	90	71	56	9
5	47	66	33	94	0	70	65	97	39	61	58	48	3	47	75	75	85	40	76	43	23	34	18	56	41	65	65	13	50	30	8	34	60	73	12	57	18	53	6	29
6	4	92	60	74	54	0	21	76	39	9	86	42	59	64	33	34	57	71	30	34	50	1	17	39	60	44	87	75	3	9	30	21	33	22	31	44	44	37	20	29
7	53	46	43	63	17	31	0	47	11	16	46	6	32	8	3	40	15	68	8	24	84	89	84	4	48	30	84	70	71	28	92	64	50	69	28	66	95	21	58	17
8	5	15	24	44	14	39	76	0	78	20	93	81	47	82	70	27	47	6	4	64	65	99	55	28	59	2	65	48	45	62	53	99	47	12	65	59	3	59	84	76
9	33	99	28	95	32	74	98	23	0	26	17	87	68	69	22	80	36	45	76	44	19	7	50	70	3	37	50	88	57	14	38	95	87	37	69	32	97	12	30	18
10	46	47	65	75	60	82	18	58	71	0	67	45	38	78	29	49	24	67	24	70	30	46	8	1	61	3	63	91	39	0	91	79	2	77	72	28	64	70	70	67
11	98	22	85	73	55	55	85	36	12	84	0	54	36	10	32	49	6	24	90	72	35	93	6	11	89	87	46	49	34	50	17	79	67	56	22	75	62	72	14	43
12	87	36	60	60	69	50	18	57	74	70	55	0	88	12	50	48	88	62	40	16	47	19	0	67	10	42	13	77	24	52	3	69	70	31	19	4	45	67	54	75
13	50	76	84	79	69	12	24	57	58	83	18	82	0	88	4	76	95	88	69	16	73	96	79	64	53	54	91	16	20	33	28	55	66	0	5	11	29	83	9	97
14	84	61	83	74	71	38	6	70	0	49	55	2	5	0	76	2	39	3	78	30	6	64	73	60	55	20	57	91	87	50	13	39	36	0	98	21	12	21	59	10
15	69	66	21	7	21	25	36	18	95	76	40	75	15	91	0	66	59	60	56	57	21	58	80	69	13	22	12	6	1	36	98	16	32	82	48	41	94	35	58	98
16	78	13	15	93	99	8	90	47	31	87	48	9	41	6	1	0	99	4	67	72	8	73	48	39	47	79	69	98	82	69	57	9	50	91	94	78	44	92	29	40
17	65	25	71	87	2	68	23	20	58	69	3	76	68	39	21	40	0	11	58	96	13	37	99	34	48	82	35	44	10	28	32	68	59	30	73	68	52	15	11	98
18	16	71	13	32	84	54	80	11	12	49	81	55	24	46	27	29	55	0	32	31	98	97	32	79	15	72	12	51	36	72	27	44	56	39	16	68	38	33	50	25
19	22	12	88	77	15	56	86	91	7	58	66	47	71	87	94	10	93	79	0	2	13	23	88	37	8	74	76	68	57	35	20	85	48	31	9	45	29	59	12	29
20	71	99	14	97	97	14	88	77	41	89	39	68	92	99	72	94	66	33	58	0	1	88	9	17	83	26	47	41	14	60	14	44	18	97	27	50	23	7	94	38
21	34	55	74	48	63	53	62	76	9	6	49	35	9	62	4	39	93	31	28	0	0	45	67	12	56	55	43	13	95	77	68	98	41	7	57	52	62	3	28	41
22	87	35	86	16	73	59	94	15	20	25	78	31	65	26	20	68	28	2	52	51	40	0	7	37	85	71	12	12	64	30	18	57	55	36	19	67	71	92	76	44
23	90	64	51	97	51	67	32	1	70	35	13	46	77	51	26	46	59	2	82	20	70	41	0	18	90	82	64	44	82	37	27	36	21	7	67	69	32	94	79	19
24	71	31	4	74	70	50	71	55	80	81	83	72	15	47	31	69	55	99	6	47	76	26	30	0	25	54	84	87	32	4	58	48	43	84	21	14	43	40	39	26
25	57	2	79	80	48	30	11	35	3	79	37	97	97	2	69	71	83	47	60	89	75	62	76	70	0	39	70	52	89	8	30	16	94	87	28	83	38	6	95	35
26	57	1	23	1	63	27	89	1	36	92	66	62	48	88	36	94	13	85	86	64	63	42	14	29	86	0	52	46	0	70	90	27	89	89	46	91	78	70	60	97
27	86	83	42	98	97	40	39	4	82	27	69	68	99	58	38	76	91	32	11	84	2	73	39	94	23	77	0	63	5	3	80	20	32	26	24	99	81	33	36	99
28	69	78	35	78	32	81	31	67	35	6	76	2	21	77	81	75	93	77	88	61	74	45	2	47	49	95	60	0	93	44	28	91	37	47	83	77	37	10	61	75
29	42	35	93	56	7	74	96	12	61	95	39	37	70	74	31	40	56	89	52	1	69	38	93	53	34	71	43	74	0	84	37	9	78	85	64	66	5	42	31	66
30	77	22	95	27	65	28	22	4	67	8	16	9	72	61	88	23	43	41	72	33	97	77	57	63	45	58	61	39	20	0	78	80	42	70	73	3	78	28	90	83
31	25	32	75	92	53	37	79	84	63	17	85	35	55	58	59	55	19	79	24	46	12	72	99	67	1	46	87	7	54	82	0	74	42	68	82	35	47	99	69	39
32	11	36	38	92	94	38	71	92	67	67	5	65	1	81	74	24	88	29	25	23	56	64	61	69	90	2	95	87	59	54	2	0	20	13	11	98	2	99	8	13
33	43	32	77	9	92	45	50	58	32	44	81	25	49	41	92	84	53	86	18	9	61	92	87	69	43	51	72	30	13	69	83	64	0	38	78	91	16	27	0	46
34	9	12	70	73	31	46	75	73	76	29	22	29	91	79	79	45	95	93	74	40	61	9	70	68	26	25	41	83	86	77	52	73	16	0	80	7	55	19	8	76
35	80	31	62	91	16	15	82	13	13	48	36	3	67	99	57	3	23	60	0	12	12	99	37	65	44	3	96	11	61	45	75	0	44	41	0	58	43	73	35	18
36	4	31	87	4	30	77	89	53	65	62	18	63	72	96	39	22	53	63	92	19	42	36	5	45	84	7	81	16	89	62	25	67	93	89	85	0	11	53	75	53
37	37	63	61	74	33	52	37	25	76	56	2	83	70	43	21	69	51	34	99	99	67	78	58	30	93	29	42	97	77	82	48	54	48	96	78	74	0	54	67	37
38	66	77	44	92	12	21	60	51	12	19	18	61	62	55	31	3	79	0	65	0	62	56	65	6	33	35	60	25	54	76	54	64	86	3	34	61	88	0	80	4
39	63	28	0	72	29	99	78	12	10	10	65	3	60	36	84	83	58	67	30	20	34	4	33	27	19	48	50	57	34	54	99	83	7	68	54	44	80	58	0	6
40	24	50	15	5	30	44	10	45	99	24	55	89	39	55	31	77	36	99	98	9	23	58	80	84	38	43	42	52	62	74	41	15	39	15	91	75	71	28	8	0

Appendix Figure A14 The amount of distance between location *j* to location *l* at period 6 (D_{jl6})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	64	47	83	59	59	12	20	58	57	71	73	99	69	96	54	51	14	72	58	12	4	94	97	63	81	59	56	18	0	17	45	30	13	75	72	46	63	66	6
2	90	0	34	97	28	42	8	87	10	63	93	71	21	5	59	48	13	47	95	47	44	23	89	65	78	89	12	88	44	46	53	72	91	28	57	75	62	34	73	52
3	53	71	0	20	44	3	39	64	3	0	58	50	70	7	81	75	48	6	83	50	95	69	9	22	11	67	54	50	35	72	10	43	9	79	81	1	81	25	50	68
4	67	68	63	0	98	70	7	27	62	23	92	45	8	99	63	72	34	66	25	63	28	55	66	45	66	98	36	82	18	62	62	5	37	31	63	2	54	48	76	61
5	86	7	30	48	0	67	35	52	5	23	14	89	54	31	92	88	98	11	3	68	62	52	86	22	53	58	11	23	46	32	85	93	39	70	18	69	71	62	37	86
6	4	1	73	28	24	0	51	23	62	24	47	90	86	61	29	35	13	77	24	96	10	77	65	30	23	93	53	90	42	50	90	22	64	53	98	0	16	12	20	88
7	81	82	68	79	39	66	0	10	68	20	0	25	74	25	10	0	66	65	54	28	83	87	71	1	81	17	74	56	77	66	24	6	7	27	38	46	30	47	28	43
8	85	42	53	86	8	17	11	0	2	25	90	45	2	99	53	50	1	12	68	96	35	86	96	11	23	89	43	50	60	94	61	27	80	59	31	97	8	40	32	58
9	98	60	86	30	40	84	17	84	0	86	10	58	65	24	64	59	67	92	13	23	64	44	5	40	80	73	76	57	96	93	70	21	7	96	83	6	10	92	45	19
10	90	56	80	62	42	10	2	82	26	0	48	68	1	12	71	58	47	51	26	89	10	53	44	37	96	86	97	88	0	65	18	70	47	77	1	20	0	33	84	75
11	80	22	36	88	60	74	4	67	85	90	0	2	70	78	72	60	88	22	7	92	14	6	8	5	13	73	31	84	24	91	74	76	81	75	92	50	86	76	46	47
12	33	1	14	23	85	7	70	93	65	92	64	0	30	40	44	60	59	77	8	72	53	64	89	35	46	1	97	0	43	73	60	96	47	59	15	0	20	58	22	84
13	39	99	33	82	63	30	1	83	85	81	24	51	0	92	67	26	17	87	37	52	92	94	98	33	6	8	9	35	44	64	55	34	19	98	43	86	80	26	30	26
14	14	99	79	93	79	97	25	37	80	5	75	6	62	0	79	77	58	88	31	60	59	15	67	32	43	90	42	6	11	35	64	39	32	45	68	34	14	20	59	27
15	6	50	72	90	96	88	9	61	2	27	79	94	11	92	0	75	46	88	27	95	37	4	81	28	73	12	44	76	52	90	95	53	23	35	5	95	86	9	64	39
16	0	29	2	82	74	82	20	61	86	65	0	21	60	41	29	0	48	75	40	96	18	19	95	46	27	80	22	13	11	48	84	74	92	10	49	68	7	5	69	71
17	22	12	60	1	84	86	19	43	4	82	84	3	28	63	76	51	0	93	33	32	58	34	40	70	63	50	94	77	96	28	48	57	7	70	39	24	11	38	37	35
18	48	82	78	35	42	14	49	32	23	69	83	38	53	35	48	22	76	0	98	59	81	77	4	61	13	41	1	55	26	26	24	26	51	90	8	37	56	21	98	6
19	96	78	58	10	82	89	91	45	11	2	88	82	66	97	3	62	7	4	0	6	41	39	48	53	81	94	25	50	44	66	58	55	30	57	26	41	65	47	53	25
20	80	21	94	80	9	15	98	68	59	59	72	77	83	34	35	70	88	98	76	0	5	53	3	76	36	15	15	24	74	52	42	1	38	31	59	96	87	53	38	76
21	21	84	37	84	60	62	49	48	85	6	49	21	82	84	24	28	78	47	62	73	0	83	74	72	85	49	45	66	61	67	68	35	57	59	27	79	3	61	83	21
22	52	14	88	84	10	84	68	26	33	6	9	47	27	2	39	32	58	37	27	15	5	0	80	59	18	41	50	63	43	66	8	69	71	89	63	0	99	32	16	13
23	54	70	1	14	73	81	76	60	57	19	46	2	9	85	49	39	16	84	6	30	74	16	0	5	4	55	67	34	11	26	45	57	85	0	72	20	19	50	62	67
24	44	70	77	29	94	89	63	40	9	88	18	68	40	59	94	72	97	2	56	33	23	66	70	0	32	87	52	2	12	45	43	31	89	8	82	70	81	91	87	89
25	98	86	85	72	15	49	61	48	30	65	25	37	85	63	51	26	58	31	53	73	80	84	92	38	0	67	68	94	65	2	77	97	3	88	92	89	37	83	37	99
26	73	2	48	94	5	11	30	65	88	39	38	54	0	18	93	52	10	14	2	53	21	16	28	87	95	0	73	77	26	24	73	87	47	92	76	46	46	35	3	13
27	7	43	38	12	0	0	68	24	19	42	76	64	36	62	96	21	84	11	34	92	79	67	2	74	3	21	0	15	84	24	90	18	74	53	17	78	79	69	2	18
28	80	54	3	13	19	81	89	36	60	63	37	66	53	90	95	61	73	84	22	67	83	73	80	15	42	62	74	0	83	43	42	28	9	16	29	18	2	47	10	95
29	90	4	4	68	95	19	97	40	65	80	77	90	52	26	92	5	6	15	71	4	22	15	0	88	53	17	73	88	0	45	16	20	54	57	14	17	12	73	96	67
30	93	69	54	18	53	97	4	95	4	81	27	16	12	12	3	68	67	67	91	18	52	54	96	44	78	5	10	89	53	0	53	17	81	79	68	88	22	38	65	17
31	9	93	35	98	60	99	10	72	19	44	65	46	15	65	38	74	85	58	74	65	10	53	27	4	43	31	51	4	5	61	0	88	76	49	92	89	98	55	70	25
32	26	63	79	68	1	78	60	37	9	71	57	56	0	8	96	29	95	90	61	32	44	41	92	4	10	65	9	30	90	94	72	0	93	69	17	44	89	42	55	32
33	96	93	80	85	41	86	59	27	63	29	21	11	21	15	61	32	47	1	67	5	5	79	90	26	24	75	99	61	22	82	11	75	0	63	62	71	89	24	58	51
34	72	35	40	98	87	8	23	3	65	80	23	68	65	59	56	25	15	66	49	52	53	86	76	75	7	92	24	70	3	57	93	31	0	0	18	21	7	89	26	61
35	5	26	55	44	91	43	13	66	36	31	46	30	83	75	64	27	10	59	26	43	57	82	28	4	23	98	74	63	57	91	25	12	17	75	0	63	3	86	82	6
36	41	9	69	30	23	53	32	72	25	43	31	51	47	38	98	24	69	21	86	24	81	39	70	57	11	9	54	52	61	68	61	87	76	74	93	0	95	62	63	66
37	93	84	92	47	64	18	98	98	44	58	43	10	57	91	85	29	94	65	41	85	31	61	56	99	5	78	34	89	47	26	20	97	56	11	68	37	0	8	70	17
38	11	82	34	88	58	49	26	61	73	89	4	84	7	98	38	40	10	10	60	16	79	46	38	58	20	16	70	71	8	79	70	89	1	26	12	82	33	0	28	29
39	45	35	56	24	43	77	55	66	94	23	20	66	67	29	12	86	46	86	16	48	63	74	67	72	68	25	32	12	71	91	49	1	27	41	24	31	63	66	0	65
40	93	74	48	48	53	10	41	62	46	42	56	42	71	94	75	61	35	30	58	92	91	61	45	64	65	0	78	66	6	74	16	21	33	94	27	27	20	35	28	0

Appendix Figure A15 The amount of distance between location *j* to location *l* at period 7 (D_{jl7})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	72	12	39	86	51	64	30	85	20	69	74	86	27	33	19	18	48	80	13	28	34	0	52	57	62	59	19	13	20	73	62	22	20	85	97	53	43	2	41
2	33	0	65	9	12	85	36	93	66	27	15	63	23	94	46	34	45	4	64	20	36	38	49	17	84	21	11	48	12	81	41	84	74	90	88	57	70	87	68	40
3	41	65	0	66	85	84	55	70	71	5	68	14	54	95	25	35	3	58	48	38	12	6	58	16	89	20	62	28	92	81	54	64	49	18	95	43	49	97	76	34
4	77	9	5	0	0	13	85	88	60	92	36	89	55	74	46	26	9	46	24	6	21	28	63	42	94	84	52	47	43	0	19	23	25	39	94	33	99	81	15	53
5	45	27	11	53	0	81	81	7	37	80	62	33	19	5	56	99	56	25	86	15	34	22	94	52	47	15	67	8	45	61	5	13	67	2	66	69	50	40	94	11
6	2	42	77	81	51	0	70	97	46	61	83	57	0	40	88	30	4	98	26	92	35	18	78	87	3	8	99	82	60	14	61	79	25	79	3	88	11	76	85	52
7	28	81	32	22	88	6	0	29	29	76	74	6	55	3	99	90	85	48	15	66	41	84	86	36	5	83	73	12	2	36	15	11	38	76	78	55	3	6	52	80
8	48	91	29	28	57	22	20	0	80	42	39	91	98	32	96	43	20	72	27	23	71	33	40	20	0	51	14	72	75	87	71	83	82	45	54	95	29	54	6	17
9	3	29	36	10	35	88	70	70	0	20	49	3	85	88	83	24	87	44	98	85	30	10	22	71	9	41	42	87	91	77	92	33	6	87	3	81	0	33	3	63
10	70	44	70	9	69	26	4	34	70	0	89	38	49	21	30	57	6	53	9	3	54	84	7	16	36	82	24	52	14	48	31	50	97	32	25	60	73	17	86	23
11	81	24	56	60	75	75	2	42	18	58	0	73	69	0	58	66	16	2	27	42	89	90	83	95	91	57	77	74	8	22	59	63	91	59	19	32	3	3	21	3
12	38	80	52	84	17	9	25	82	76	45	38	0	82	41	41	7	27	17	74	52	11	48	78	0	6	72	26	75	33	9	22	32	6	83	37	92	70	33	58	5
13	20	31	55	38	79	37	64	58	93	38	6	76	0	12	37	84	58	36	97	67	75	55	9	84	35	2	71	31	80	44	38	82	14	30	90	16	0	83	57	53
14	78	20	14	41	38	57	91	73	37	47	49	29	72	0	83	46	49	25	36	66	92	28	45	29	56	99	40	58	39	71	46	86	38	74	98	46	80	22	17	71
15	46	93	34	38	37	85	13	35	1	26	2	91	19	59	0	92	21	78	73	47	13	24	35	4	60	18	1	25	36	40	94	41	20	4	65	60	4	54	79	35
16	52	6	60	63	28	37	40	91	64	1	58	30	8	79	28	0	27	69	7	97	79	7	17	3	15	72	19	83	30	65	44	93	36	65	31	56	1	61	77	79
17	46	98	40	90	78	43	37	77	7	71	45	31	74	19	20	87	0	23	74	83	62	36	13	12	60	69	99	25	14	32	45	22	44	35	22	96	6	18	42	23
18	70	20	16	40	26	25	14	19	66	68	14	34	95	1	56	65	11	0	72	49	72	97	99	56	7	89	16	47	34	91	49	43	22	26	47	29	36	29	51	45
19	16	41	4	31	15	2	23	95	55	24	42	24	8	6	72	92	3	3	0	0	55	52	63	61	97	91	19	48	29	26	84	68	40	99	27	0	90	48	77	28
20	94	31	65	17	7	34	11	30	29	45	86	64	2	48	12	47	96	90	62	0	98	29	88	77	37	78	45	23	15	25	77	46	99	86	54	43	59	36	4	67
21	85	17	48	73	53	85	30	30	3	81	87	14	36	87	18	39	70	15	76	25	0	71	44	39	8	37	1	41	16	28	57	55	65	43	18	34	4	83	27	10
22	79	33	69	71	11	41	94	18	99	27	17	64	50	89	39	82	41	60	41	94	70	0	95	21	6	66	97	10	12	91	86	99	63	36	79	6	78	76	10	48
23	31	12	3	42	79	94	99	32	5	91	62	1	40	49	2	1	32	73	1	19	68	79	0	3	17	76	17	46	22	57	37	85	80	75	67	55	96	9	90	35
24	36	98	5	59	29	75	78	72	11	49	21	69	34	54	71	48	85	17	12	3	11	53	80	0	66	28	37	65	66	25	76	28	47	84	49	25	52	66	64	46
25	82	91	44	37	2	49	77	41	24	55	24	97	92	41	68	30	59	96	43	82	65	54	69	17	0	77	7	42	12	79	75	19	38	70	64	40	72	51	14	53
26	94	3	36	7	13	93	54	12	63	37	64	43	22	87	63	90	90	61	55	28	52	83	39	99	39	0	8	98	70	71	77	49	93	91	9	90	75	55	20	15
27	45	95	45	46	32	31	15	24	27	78	97	45	62	36	66	52	41	33	76	2	12	57	68	25	13	82	0	27	42	71	2	74	63	38	74	54	70	12	71	51
28	93	10	75	4	83	91	62	54	36	36	8	6	75	53	69	45	77	20	27	80	13	22	40	6	56	57	76	0	67	94	35	12	89	20	5	19	54	98	20	84
29	31	26	59	35	57	19	87	57	26	28	68	60	28	97	67	88	11	6	48	79	99	56	30	80	61	31	86	83	0	24	84	55	40	2	0	95	41	10	75	50
30	32	94	93	46	90	91	97	2	30	44	55	87	54	94	0	72	8	59	61	61	47	67	34	18	67	81	29	44	31	0	21	80	91	17	26	32	3	24	60	73
31	58	52	94	62	71	71	34	61	11	4	78	93	13	51	52	23	82	23	47	98	56	43	7	52	71	12	15	70	29	21	0	86	82	87	80	1	63	91	64	81
32	49	17	63	84	8	33	10	62	61	60	62	5	60	83	80	45	25	4	4	34	22	2	27	83	0	1	40	12	79	40	22	0	61	82	26	61	37	33	92	22
33	61	42	42	43	69	99	44	98	61	30	49	58	43	80	23	88	86	61	44	30	65	13	8	13	99	15	60	26	1	57	13	31	0	22	78	2	39	7	71	43
34	61	14	32	62	31	58	68	41	57	89	10	43	46	20	99	68	73	60	88	45	4	1	24	49	96	23	93	25	29	86	16	19	15	0	77	3	92	70	9	53
35	81	33	21	87	67	79	99	26	86	83	83	81	3	98	62	70	68	22	81	75	82	54	28	9	66	15	41	31	37	14	6	25	95	82	0	21	98	76	23	45
36	49	48	15	33	26	42	67	49	75	3	48	11	11	80	92	8	4	11	36	18	87	75	16	34	82	29	89	57	72	19	3	7	77	23	46	0	57	39	15	2
37	72	38	60	46	11	98	24	46	35	48	2	98	28	35	91	19	62	73	70	43	21	20	86	26	15	67	36	83	66	73	15	96	2	62	94	27	0	12	31	68
38	12	43	2	56	40	6	90	61	45	14	28	78	32	49	82	32	42	58	85	36	99	18	18	73	76	90	30	65	8	44	77	15	35	62	86	49	76	0	5	41
39	11	40	64	8	51	81	76	16	95	73	94	86	81	22	19	33	98	56	14	92	23	83	59	68	53	79	94	17	6	9	40	52	9	45	1	14	52	62	0	55
40	1	50	90	24	43	56	60	50	97	74	71	96	7	7	55	62	95	85	9	1	91	9	15	15	95	61	95	42	77	0	99	13	99	43	74	31	13	56	76	0

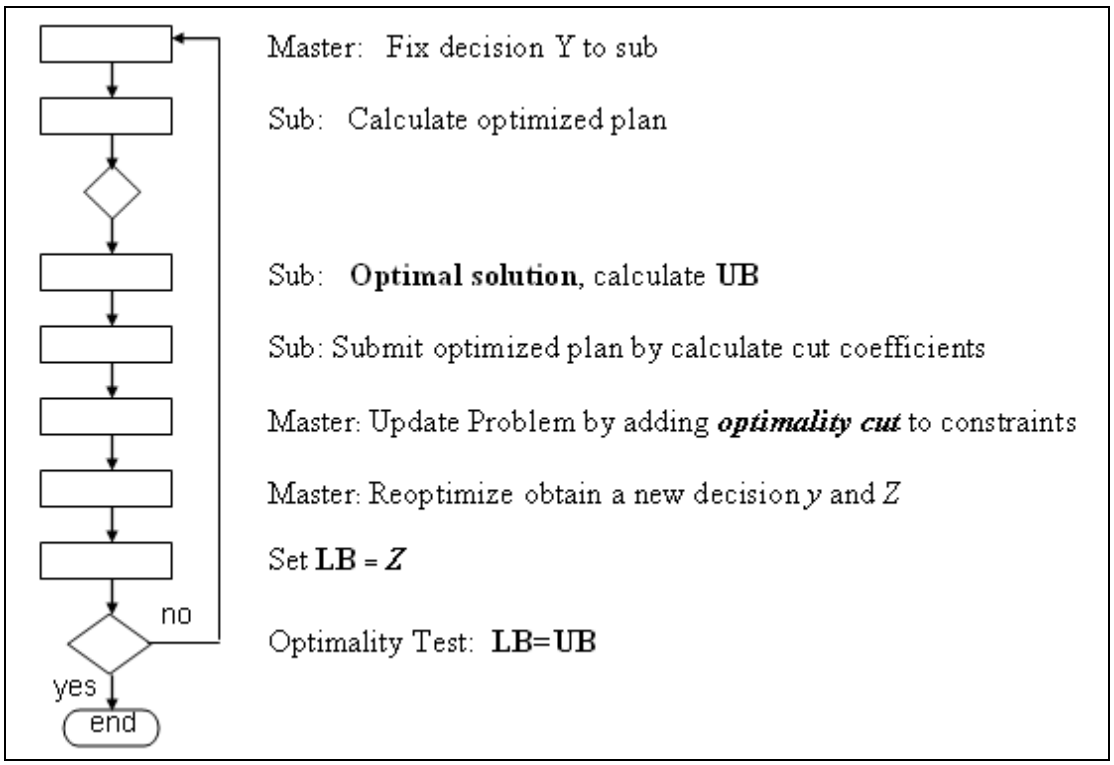
Appendix Figure A16 The amount of distance between location *j* to location *l* at period 8 (D_{jl8})

j\l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	38	74	89	71	39	66	42	78	87	94	5	3	55	43	65	45	16	47	73	97	15	25	49	4	56	58	89	53	48	22	68	14	90	56	48	89	43	51	30
2	38	0	62	20	50	41	62	44	35	65	0	80	70	69	29	15	13	12	75	79	38	3	4	71	17	18	90	51	98	30	60	12	37	35	91	91	65	22	99	70
3	74	16	0	32	57	42	5	72	40	95	15	0	95	75	13	5	95	96	77	56	0	70	84	59	3	90	21	7	85	77	5	79	87	47	48	19	10	11	79	50
4	89	78	6	0	20	3	48	48	93	17	7	98	9	98	59	67	38	72	87	10	95	93	41	59	21	19	31	1	99	89	0	96	59	84	75	68	53	34	55	36
5	71	0	71	57	0	8	61	70	23	82	13	35	11	12	55	73	1	97	9	91	51	63	11	61	81	15	99	78	4	95	15	70	48	67	1	74	30	87	89	4
6	39	49	34	23	90	0	46	51	2	28	98	0	6	22	46	63	36	25	18	25	7	59	35	6	68	1	39	30	32	12	61	32	11	89	7	77	28	41	99	19
7	66	8	34	1	46	63	0	21	31	87	35	15	5	69	50	38	26	68	47	86	23	0	87	22	50	39	11	15	52	93	47	99	38	12	38	27	7	47	33	4
8	42	57	97	90	91	0	53	0	0	96	1	95	53	72	58	85	13	60	1	45	66	39	3	74	34	29	57	54	31	88	98	0	60	72	40	8	90	95	5	51
9	78	71	0	16	9	13	81	38	0	43	54	39	65	26	60	7	74	76	73	9	95	78	89	52	41	15	80	39	47	36	99	47	58	89	33	73	51	95	43	59
10	87	81	37	49	98	67	35	84	65	0	39	63	0	48	12	28	82	0	89	55	47	11	5	37	32	11	40	61	26	67	36	29	89	32	43	26	63	84	77	85
11	94	12	99	78	60	46	45	81	43	63	0	27	80	43	5	61	64	66	64	40	76	90	0	86	88	15	59	51	25	25	9	20	56	95	37	2	88	29	50	34
12	5	78	84	20	49	40	52	12	77	7	54	0	99	29	12	84	96	32	59	0	99	45	16	88	40	63	41	52	28	49	43	9	0	34	97	63	17	93	35	22
13	3	75	44	0	1	93	43	72	65	7	83	83	0	44	22	11	67	50	15	5	23	10	20	36	48	7	94	52	92	97	94	97	68	18	9	47	76	9	23	33
14	55	84	46	63	43	51	32	43	37	52	24	98	71	0	86	62	44	85	6	93	43	82	13	62	21	24	34	40	80	40	29	67	13	34	44	63	32	9	36	37
15	43	49	98	88	58	54	58	73	19	90	4	56	76	42	0	61	26	69	60	21	38	82	34	0	9	91	45	51	66	8	98	20	10	98	19	65	44	65	32	2
16	65	99	34	78	19	97	8	35	20	30	50	88	96	36	26	0	74	70	25	45	95	78	77	81	29	42	11	24	26	94	35	10	37	0	81	7	65	44	14	18
17	45	80	1	1	0	35	57	60	19	96	30	82	83	15	12	11	0	0	14	18	3	92	34	41	60	8	59	62	12	39	35	21	23	92	78	0	50	6	38	18
18	16	81	14	20	81	28	9	66	99	15	17	40	23	47	0	10	25	0	91	66	79	53	98	78	63	44	80	98	14	0	10	68	11	34	73	96	28	40	84	10
19	47	95	70	45	97	7	86	87	10	84	5	4	3	91	47	12	44	93	0	44	22	59	66	41	0	28	11	49	92	30	35	6	22	35	68	19	36	50	94	86
20	73	26	12	1	66	2	84	17	27	9	24	64	24	46	23	7	34	82	83	0	68	71	14	13	31	80	56	11	0	86	7	7	45	19	0	12	6	70	10	89
21	97	42	15	61	63	0	39	20	19	72	99	91	93	29	79	66	80	42	9	3	0	72	79	27	70	85	32	9	25	93	73	11	50	27	73	56	53	84	74	47
22	15	44	85	84	84	82	46	98	47	93	66	47	0	32	8	0	35	70	38	93	90	0	3	6	59	5	85	38	21	18	54	17	73	91	27	6	46	65	89	5
23	25	10	10	39	32	18	29	72	51	26	26	55	24	60	63	45	90	18	65	43	17	24	0	48	85	0	87	21	31	23	19	11	37	95	34	36	1	46	93	74
24	49	76	31	39	48	95	67	13	81	94	52	51	3	90	52	81	45	79	32	22	3	49	45	0	26	84	22	15	52	77	5	11	81	84	66	0	46	97	64	53
25	4	68	97	51	83	0	0	51	86	91	65	57	45	93	30	55	53	13	58	3	81	92	13	78	0	88	41	5	59	36	46	44	35	53	92	57	98	52	80	15
26	56	96	50	36	68	82	19	31	3	20	44	66	49	81	6	86	0	76	99	63	40	83	48	37	34	0	32	58	42	88	40	68	11	36	62	13	42	16	88	99
27	58	21	67	31	89	34	23	59	49	48	81	4	42	55	53	52	91	54	31	31	63	49	36	45	23	12	0	12	64	49	84	80	37	18	53	3	35	89	2	7
28	89	18	38	25	97	77	27	62	68	1	59	88	85	2	73	8	36	26	74	4	48	62	52	96	50	83	6	0	9	45	74	4	19	33	50	82	0	82	51	52
29	53	81	58	27	8	54	93	0	44	51	60	56	32	0	70	34	8	23	60	61	64	73	95	99	21	37	12	27	0	74	16	86	49	30	12	84	93	17	39	22
30	48	59	15	67	33	10	82	10	65	49	84	35	92	57	38	93	69	0	44	91	31	15	24	84	6	85	92	62	35	0	69	58	44	36	18	51	78	77	21	10
31	22	95	23	58	97	94	34	24	43	81	6	98	78	78	61	67	5	33	50	10	14	39	17	10	42	48	60	0	5	37	0	52	50	26	42	73	12	52	17	44
32	68	83	16	89	20	26	78	7	25	21	6	68	53	60	62	51	41	92	58	12	41	93	93	87	52	45	31	59	77	94	82	77	41	13	59	93	60	0	43	97
33	14	61	39	58	7	8	51	75	0	40	49	36	65	68	78	37	7	67	81	84	40	3	2	40	27	67	51	48	35	40	86	33	0	73	7	28	2	20	15	68
34	90	37	79	39	55	10	68	83	91	67	32	48	23	75	93	78	83	25	0	46	41	66	92	18	28	31	67	59	20	23	14	44	74	0	40	22	83	89	42	68
35	56	85	79	2	3	80	1	5	77	3	50	21	67	12	92	45	61	69	41	56	50	29	96	47	14	74	71	5	0	23	53	71	74	23	0	18	89	30	34	8
36	48	87	40	74	37	76	50	93	45	93	19	35	37	81	20	1	81	42	65	19	46	62	26	55	60	33	22	42	58	52	67	4	82	22	71	0	40	56	0	46
37	89	2	36	26	82	76	82	30	60	0	14	40	46	90	19	21	23	10	5	82	60	15	71	76	38	63	1	33	36	53	42	18	13	70	82	44	0	74	10	51
38	43	9	55	33	94	47	88	87	93	87	69	23	6	4	81	63	61	12	92	0	15	57	70	23	18	99	61	54	90	43	53	10	90	91	14	12	42	0	57	87
39	51	60	1	65	9	49	95	26	21	31	52	5	44	3	17	98	90	24	12	45	76	1	32	7	26	35	18	55	85	0	86	25	27	78	80	12	83	33	0	84
40	30	1	43	18	5	46	92	51	13	24	46	98	98	45	46	41	48	61	95	31	68	0	19	93	54	55	1	23	62	73	72	92	68	45	5	63	13	14	95	0

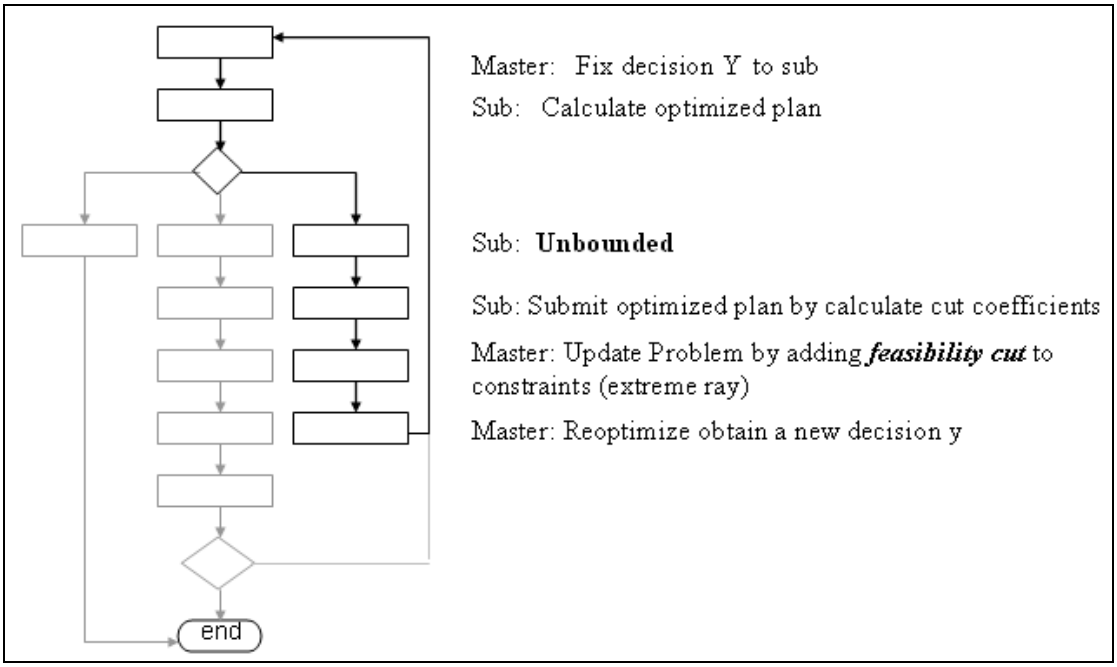
Appendix Figure A17 The rearrangement cost when facility *i* assigned to location *j* at period *t* and move to location *l* period *t*+1 (R_{ijlt})

APPENDIX B

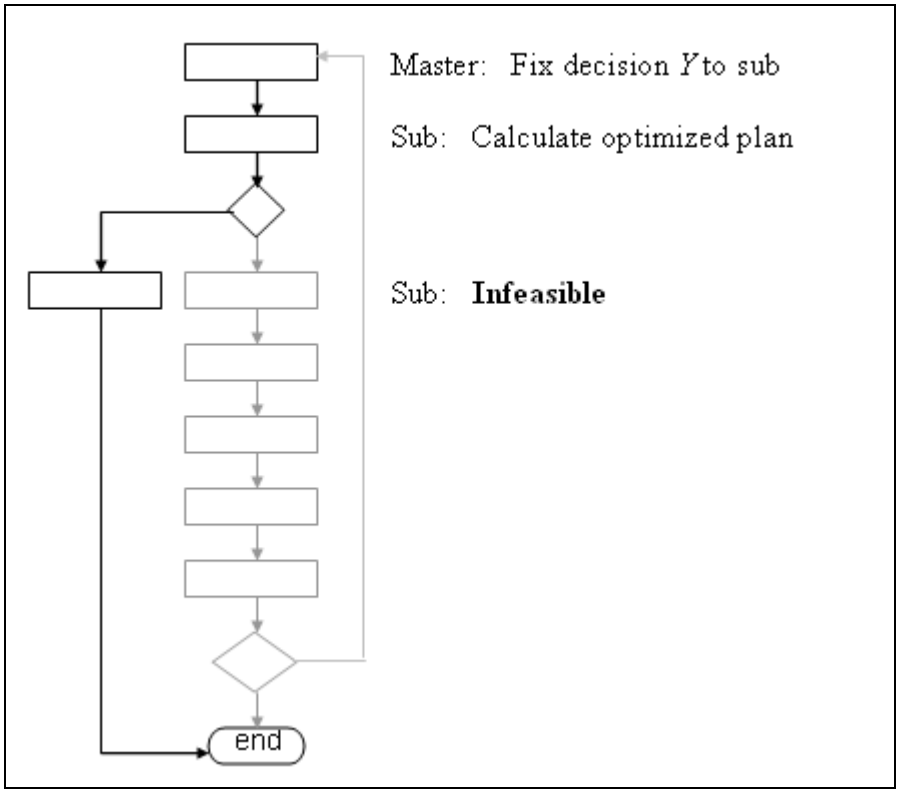
Benders' decomposition Algorithm Flowchart



Appendix Figure B1 The flowchart of Benders decomposition (Case: Optimal)



Appendix Figure B2 The flowchart of Benders decomposition (Case: Unbounded)



Appendix Figure B3 The flowchart of Benders decomposition (Case: Infrasible)

APPENDIX C

MATLAB Source Code

MATLAB Source code for ABD+ADP with Trust Region (Successive Adaptation Procedure)

Main.m

```
%%%
%
%   Combinatorial Optimisation Method + Successive Adpation Procedure
%   1. Bender's decomposition (approx - LP +
%      Hungarian   method for mixed-integer problem in master problem)
%   2. Trust region constraint for master problem
%   3. Dynamic Programming
%
%   Problem: Dynamic Quadratic Assignment Problem
%
%   28 Auguest 2008
%
%   Comments:
%      1. Carry over layout from previous iterations to do DP in the next
%         iteration
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
diary log.out

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs

% File name
filename = 'FDG.data.xls';

% Problem size
I  = 6; J  = 6;
K  = I; L  = J;
T  = 3;

%
% Parameters for the combinatorial optimisation
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Parameters for successive adpation procedure
%      in Benders Decomposition

% Number of cuts
Ncut_max = [10 20 40 80];
```

```
Ncut_max_step = length(Ncut_max);

% Trust Region Size
TR_size_constant = [1 0.5 0.25];
TR_size_step = length(TR_size_constant);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Maximum number of iteration - Bender's Decomp + DP loop
iter_max = 20;

% Tolerance for stopping criteria - Bender's Decomp + DP loop
TC_Tol = 1e-5;

%
% Parameters for Bender's Decomposition **** very critical ****
%

% Stopping Criteria
%iflag_stop = 0; % stop when LB == UB (Exact!)
iflag_stop = 3; % stop when LB > mean(z_sub) - 3 std(z_sub)
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   1. Flow between facility i to k at t: Fikt
Fikt = zeros(I,K,T);
Fikt(:,:,1) = xlsread(filename,'F1','A1:F6');
Fikt(:,:,2) = xlsread(filename,'F2','A1:F6');
Fikt(:,:,3) = xlsread(filename,'F3','A1:F6');
%Fikt(:,:,4) = xlsread(filename,'F4','A1:F6');
%Fikt(:,:,5) = xlsread(filename,'F5','A1:F6');
%Fikt(:,:,6) = xlsread(filename,'F6','A1:F6');
%Fikt(:,:,7) = xlsread(filename,'F7','A1:F6');
%Fikt(:,:,8) = xlsread(filename,'F8','A1:F6');

%   2. Distance between location j to l: Djl
Djlt = zeros(J,L,T);
Djlt(:,:,1) = xlsread(filename,'D1','A1:F6');
Djlt(:,:,2) = xlsread(filename,'D2','A1:F6');
Djlt(:,:,3) = xlsread(filename,'D3','A1:F6');
%Djlt(:,:,4) = xlsread(filename,'D4','A1:F6');
%Djlt(:,:,5) = xlsread(filename,'D5','A1:F6');
%Djlt(:,:,6) = xlsread(filename,'D6','A1:F6');
%Djlt(:,:,7) = xlsread(filename,'D7','A1:F6');
```

```
%Djlt(:,:,8) = xlsread(filename,'D8','A1:F6');

%  3. G - rearrangement cost
Gjlt = zeros(J,L,T-1);
Gjlt(:,:,1) = xlsread(filename,'Gab','A1:F6');
Gjlt(:,:,2) = xlsread(filename,'Gab','A1:F6');
%Gjlt(:,:,3) = xlsread(filename,'Gab','A1:F6');
%Gjlt(:,:,4) = xlsread(filename,'Gab','A1:F6');
%Gjlt(:,:,5) = xlsread(filename,'Gab','A1:F6');
%Gjlt(:,:,6) = xlsread(filename,'Gab','A1:F6');
%Gjlt(:,:,7) = xlsread(filename,'Gab','A1:F6');

%   4. Initial (guess) assignment: xij0 (with I facilities and J locations)
%      4.1 Generate sutomatically Initial L/O
xijt0 = zeros(I,J,T);
for itime = 1:T
   xijt0(:,:,itime) = eye(I);
end
 %xijt0 = zeros(I,J,T);
%for t=1:T
%    for i = 1:I
%        xijt0(i,ind0(t,i),t) = 1;
%    end
%end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Problem Constriants

% Define cost of assignment at t: Cijklt = Fikt * Djlt
Cijklt = zeros(I,J,K,L,T);
for t = 1:T
   for l = 1:L
      for k = 1:K
         Cijklt(:,:,k,l,t) = Fikt(:,k,t)*Djlt(:,l,t)';
      end
   end
end
% Define rearrange cost from t to t+1: Rijl(t+1) = Gjlt for facility ith
for i = 1:I
   Rijltp(i,:,:,1:(T-1)) = Gjlt(:,:,1:(T-1));
end

%   Define equality constraints - for a Master Problem
%   For each time
Asub = [];
```

```
%   Sum over each row/facility is one
for i = 1:I
   Atemp = zeros(I,J);
   Atemp(i,:) = ones(1,J);
   Asub = [Asub; reshape(Atemp,1,I*J)];
end

%   Sum over each column/location is one
for j = 1:J
   Atemp = zeros(I,J);
   Atemp(:,j) = ones(I,1);
   Asub = [Asub; reshape(Atemp,1,I*J)];
end

%   remove the last equality constraint due to its dependency on the other
%   equality constraints
Asub = Asub(1:(I+J-1),:);

Aeq = Asub;
%   For all time
for t = 2:T
   Aeq = blkdiag(Aeq,Asub);
end

%   The sum is equal to one
beq = ones((I+J-1)*T,1);

% Start the procedure

disp('#########################################################################
###')
disp('')
disp(['Start the Successive Adaptation Procedure'])
disp(['Ncut_max = ' num2str(Ncut_max)])
disp(['TR_size_constant = ' num2str(TR_size_constant)])
disp('')
disp('#########################################################################
###')
disp('')

tic     % Clock starts

%   Perform over Ncut_max
for incut = 1:Ncut_max_step

   %   Perform over TR_size_constant
   for itr = 1:TR_size_step
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %
      % Perform optimisation: Bender's decomposition (approx - LP
      %   + Hungarian method for mixed-integer problem in master problem)
      %   + Dynamic Programming
      %


disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
      disp(['Ncut_max = ' num2str(Ncut_max(incut))])
      disp(['TR_size_constant = ' num2str(TR_size_constant(itr))]);

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')

      [TC_opt,xijt_opt] = BendersTR_DP(xijt0,Cijklt,Rijltp,Aeq,beq,...
         Ncut_max(incut),TR_size_constant(itr),iter_max,TC_Tol,...
         iflag_stop);

      % Update layout
      xijt0 = xijt_opt;

      disp(' ')

   end
end

toc    % Clock stops

diary off
```

BendersTR_DP.m

```
function [TC_opt,xijt_opt] = BendersTR_DP(xijt0,Cijklt,Rijltp,Aeq,beq,...
   Ncut_max,TR_size_constant,iter_max,TC_Tol,iflag_stop)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Combinatorial optimisation - Bender's decomposition (approx - LP +
%      Hungarian   method for mixed-integer problem in master problem)
%      + Trust Region Constraint + Dynamic Programming
%
```

```
%   Problem: Dynamic Quadratic Assignment Problem
%
%   28 August 2008
%
%   Comments:
%       1. Carry over layout from previous iterations to do DP in the next
%          iteration
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Determine the size of the problem
I = size(xijt0,1); J = size(xijt0,2); % size of xij
K = I; L = J;
T = size(xijt0,3);

% Database of the layout for dynamic programming
xijt_data = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Perform optimisation: Bender's decomposition (approx - LP + Hungarian
% method for mixed-integer problem in master problem) + Dynamic Programming
%

% Find total cost ( = assignment cost + rearrangement cost) for an initial
%   guess layout, xijt0

AC = zeros(1,T);
RC = zeros(1,T);

for t = 1:T-1
    AC(t) = AssignmentC(Cijklt(:,:,:,:,t),xijt0(:,:,t));
    RC(t) = RearrangementC(Rijltp(:,:,:,t),xijt0(:,:,t),xijt0(:,:,t+1));
end
AC(T) = AssignmentC(Cijklt(:,:,:,:,T),xijt0(:,:,T));

TC_old = sum(AC) + sum(RC);

disp('')
disp(['The total cost of the initial layout = ' num2str(TC_old)])
disp('')

% Start the iteration

for iter = 1:iter_max
```

```
   disp('----------------------------------------------------------------')
   disp(['Iteration of the combinatorial optimisation = ' num2str(iter)])
   disp('----------------------------------------------------------------')

%    disp(' ')
   disp('*** Benders Decomposition ***')

   % Benders' decomposition
   [xijt_ans,total_cost,xijt,...
      bendersLB,bendersUB,benders_zsub,benders_zmaster] = ...
      benders_dqap(Cijklt,Rijltp,Aeq,beq,xijt0,Ncut_max,iflag_stop,...
      TR_size_constant);

   disp(['     Number of cuts = ' num2str(length(bendersLB))])
   disp(['     Lower and upper bounds: ' num2str([max(bendersLB)
min(bendersUB)])])
   disp(['     mean(z_sub)-3 sigma(z_sub): ' num2str(mean(benders_zsub)-
3*std(benders_zsub))]);
   disp(['     mean(z_sub): ' num2str(mean(benders_zsub))]);
   disp(['     sigma(z_sub): ' num2str(std(benders_zsub))]);

%    disp(' ')
   disp('*** Dynamic Programming ***')

   % Dynamic programming
   [TC_opt,xijt_opt,xijt_data] = dp(Cijklt,Rijltp,I,J,T,xijt,xijt_data);
   disp(['     Number of layout samples ' num2str(size(xijt_data,3))])

   % Find index of optimal layout
   loind_opt = layout_index(xijt_opt);

   % Total cost reduction
   DTC = abs((TC_opt - TC_old)/TC_opt)*100;

   disp(['     The best total cost = ' num2str(TC_opt)])
   disp(['     The best layout:'])
   for t = 1:T-1
      disp(['     ' num2str(loind_opt(t,:)) ', '])
   end
   disp(['     ' num2str(loind_opt(T,:))])
   disp(['     Reduction in the total cost from the previous iteration (%): '
num2str(DTC)])

   % Convergence check
   if DTC < TC_Tol
      disp(' ')
disp('==========================================================')
```

```
        disp(['Optimisation has converged at iteration ' num2str(iter)])
        disp(['     Optimal total cost = ' num2str(TC_opt)])
        disp('     Optimal layout = ')
        for t = 1:T-1
           disp(['     ' num2str(loind_opt(t,:)) ', '])
        end
        disp(['     ' num2str(loind_opt(T,:)) '.'])

disp('=====================================================')
        return
    end

    % Update new layout and TC_old
    xijt0 = xijt_opt;
    TC_old = TC_opt;

end

% End of iteration

if DTC > TC_Tol
    disp(' ')
    disp('============================ WARNING !
============================')
    disp(['Iteration for the combinatorial optimisation has reached a maximum iteration
' num2str(iter_max)])
    disp(['     The best total cost = ' num2str(TC_opt)])
    disp('     The best layout = ')
    for t = 1:T-1
        disp(['     ' num2str(loind_opt(t,:)) ', '])
    end
    disp(['     ' num2str(loind_opt(T,:)) '.'])

disp('=====================================================')
end
```

AssignmantC.m

```
function AssCost = AssignmentC(Cijkl,xij)

I = size(xij,1); K = I;
J = size(xij,2); L = J;

AssCost = 0;

for l = 1:L
    for k = 1:K
        for j = 1:J
```

```
            for i = 1:I
                AssCost = AssCost + Cijkl(i,j,k,l)*xij(i,j)*xij(k,l);
            end
        end
    end
end
```

RearrangementC.m

```
function RearrCost = RearrangementC(Rijltp,xij,yil)

I = size(xij,1);
J = size(xij,2); L = J;

RearrCost = 0;

for l = 1:L
    for j = 1:J
        for i = 1:I
            RearrCost = RearrCost + Rijltp(i,j,l)*xij(i,j)*yil(i,l);
        end
    end
end
```

benders_dqap.m

```
function [xijt0,z_master,xijt,...
    bendersLB,bendersUB,benders_zsub,benders_zmaster] = ...
    benders_dqap(Cijklt,Rijltp,Aeq,beq,xijt0,Ncut_max,iflag_stop,...
    TR_size_constant)

% Bender's decomposition for dynamic quadratic assignment problem

% Determine the size of the problem
I = size(xijt0,1); J = size(xijt0,2); % size of xij
K = I; L = J;
T = size(xijt0,3);

% Check the stopping criteria
if nargin < 7
    iflag_stop = 0; % stop when UB =LB (exact!)
end

% Check the number of maximum cut
if nargin < 6
    Ncut_max = factorial(sqrt(I*J))^T;
end
```

```
%%%
% Initialisation
%   Assignment
xijt(:,:,:,1) = xijt0;

%   Total assignment cost
ass_cost(1)= -Inf;

%   Bounds
LB = -Inf; UB = Inf;

%   Define equality constraints for a Master Problem - add column for z
Aeq = [Aeq zeros((I+J-1)*T,1)];

%   Cut
A_master = [];
b_master = [];
%%%

%%%
% Start iteration
for iter = 1:Ncut_max
   % Subproblem
   % This version we use benders_sub_nosimplex - don't use SIMPLEX
   [Uijklt,Vijltp,z_sub] = benders_sub(xijt0,Cijklt,Rijltp);
   if (z_sub < UB)
      UB = z_sub;
%%%%%%%%%%%%%%%%%%%%%%%%%%%% added: 1 line
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      save benders_upper.mat xijt0 UB
   end

   % Accelerating Benders Decomposition - Trust Region Constraint
   [A_TR,b_TR] = TR_constraint(xijt0,iter,Ncut_max,TR_size_constant);

   % Master Problem - Approximate
      % LP-Hungarian Method option
   [xijt0,z_master,A_master,b_master] = ...
      benders_master(Uijklt,Vijltp,A_master,b_master,Aeq,beq,A_TR,b_TR);

   if (z_master > LB)
      LB = z_master;
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%% added: 2 lines
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % show progress of the Benders decomposition
```

```
   disp(['         BD is at cut: ' num2str(iter) '    &   LB/UB: ' num2str([LB  UB])])

   % Update assignment layout and the total assignment cost
   xijt(:,:,:,iter+1) = xijt0;
   ass_cost(iter+1) = z_sub;

   % Update bounds
   bendersLB(iter) = LB;
   bendersUB(iter) = UB;
   benders_zsub(iter) = z_sub;
   benders_zmaster(iter) = z_master;

%   % Show progress
%   disp('****************************************')
%   disp(['number of cuts = ' num2str(iter)])
%   disp(['z_master: ' num2str(z_master) ...
%      ' Lower/Upper Bounds: ' num2str([LB UB])])
%   disp(['average(z_sub) - 3 sigma(z_sub): ' ...
%      num2str(mean(benders_zsub) - 3*std(benders_zsub))])

   % Stop?
   switch iflag_stop
      case (0)    % Optimality test - if UB = LB? (Exact!)
         if UB == LB
%            disp('****************************************')
            disp('     Optimal!')
%            disp(['Number of Cuts ' num2str(iter)])
%            disp('Cost Value of Master Problem')
%            disp(num2str(z_master))
%            disp('****************************************')
            return
         end

      case (1)
         disp('iflag_stop = 1; stop')
         pause

      case (3)
         LB_cutoff = mean(benders_zsub) - 3*std(benders_zsub);
         if LB >= LB_cutoff
%            disp('****************************************')
            disp('     LB >= mean(z_sub) - 3 sigma (z_sub)')
%            disp(['LB and LB_cutoff = ' num2str([LB LB_cutoff])])
%            disp(['Number of Cuts ' num2str(iter)])
%            disp('Cost Value of Master Problem')
%            disp(num2str(z_master))
%            disp('****************************************')
```

```
            return
        end

    end

end
%%%

% Warning - Reached Ncut_max
%disp('****************************************')
%disp('     Reached the maximum number of cuts. Terminated.')
%disp(['     Number of Cuts ' num2str(iter)])
%disp(['     ' A_master(:,1:I*J*T) b_master])
%disp('     Cost Value of Master Problem')
%disp(['     ' num2str(z_master)])
%disp('****************************************')
disp('     Reached the maximum number of cuts')
```

benders_sub.m

```
function [Uijklt,Vijltp,z_sub] = benders_sub(xijt,Cijklt,Rijltp)

% Benders Decomposition Subproblem - Assignment Problem
% Given xijt - facility i assigned to location j at time t;
%   Determine Uijklt - Dual Problem

%
% max c_sub * Uijklt + g_sub * Vijl(t+1)
% s.t. Uijklt >= 0,
%      Vijl(t+1) >= 0;
%      0 <= Uijklt <= Cijklt
%      0 <= Vijl(t+1) <= Rijl(t+1)
%

% Simplex is not used in this version. We simply use intuition to maximise
%   cost function i.e.
%   if (xik + xjl - 1) = 1 -> Uijklt = Cijklt (maximum value)
%   else                   -> Uijklt = 0
%   end
%
% The result of this is faster and less memory storage requirement!
%

% Problem dimension
I = size(xijt,1); J = size(xijt,2); % size of xij
K = I; L = J;
T = size(xijt,3);
```

```
% Define cost function
c_sub = -ones(I,J,K,L,T);
for t = 1:T
   for l = 1:L
      for k = 1:K
         c_sub(:,:,k,l,t) = c_sub(:,:,k,l,t) ...
            + xijt(:,:,t) + xijt(k,l,t);
      end
   end
end
c_sub = reshape(c_sub,1,I*J*K*L*T);

g_sub = -ones(I,J,L,T-1);
for t = 1:T-1
   for l = 1:L
      for i = 1:I
         g_sub(i,:,l,t) = g_sub(i,:,l,t) ...
            + xijt(i,:,t) + xijt(i,l,t+1);
      end
   end
end
g_sub = reshape(g_sub,1,I*J*L*(T-1));

%   total cost for subproblem = cost + rearrangement
c_subt = [c_sub g_sub];
Nsize = length(c_subt);

%%%
% DON'T NEED
% Constraints for both Uijklt and Vijlt
%A = speye(Nsize);
%b = zeros(Nsize,1);
%%%

% Bounds
% Lower Bound, 0
LB = zeros(Nsize,1);
% Upper Bound, Cijkl = Fik*Djl - cost of assigning facility i to ...
%   location j and facility k to location l
UB = [reshape(Cijklt,I*J*K*L*T,1); ...
      reshape(Rijltp,I*J*L*(T-1),1)]';

%%%
% DON'T NEED
% Linear Programming
%options = optimset('display','off','Largescale','off','Simplex','on');
%[Uijklt,z_sub] = linprog(-c_subt,-A,b,[],[],LB,UB,[],options);
```

```
% z_sub = -z_sub;
%%%
Uijklt = max(0,sign(c_subt)).*UB; Uijklt = Uijklt(:);
z_sub = c_subt*Uijklt;

Vijltp = Uijklt((I*J*K*L*T+1):Nsize);
Uijklt = Uijklt(1:I*J*K*L*T);

Vijltp = reshape(Vijltp,I,J,L,(T-1));
Uijklt = reshape(Uijklt,I,J,K,L,T);
```

TR_constraint.m

```
function [A_TR,b_TR] = TR_constraint(xijt0,iter,Ncut_max,TR_size_constant)

% Construct Trust-Region constraint to accelerate Benders Decomposition

% Determine the size of the problem
I = size(xijt0,1); J = size(xijt0,2); % size of xij
K = I; L = J;
T = size(xijt0,3);

% Determine the size of trust region
xiter = iter/Ncut_max;
TR_size = TR_size_constant;     % constant
%TR_size = min(TR_size_constant*(xiter+1),1); % linear increase from 0 to 1
%TR_size = min(TR_size_constant*(xiter.^8+1),1);  % quad
%xneg = xiter - 1;
%TR_size = TR_size_constant*(exp(-(6*xneg).^2)+1);

% Add trust region constraint i.e.
%   sum_existing (1-xijt) + sum_rest (xijt) <= ?*N*T

A_TR = [ones(1,I*J*T) 0] - 2*[reshape(xijt0,1,I*J*T) 0];
b_TR = TR_size*I*T - I*T;
```

benders_master.m

```
function [xijt,z_master,A_master,b_master] = ...
    benders_master(Uijklt,Vijltp,A_master,b_master,Aeq,beq,A_TR,b_TR)

% Benders Decomposition Master problem (approximate) - Assignment Problem
% Given Uijkl - solution from subproblem (dual problem)
%       A_master and b_master - previous cuts
%   Determine xij - new assignment: facility i assigned to location j
%           A_master and b_master - include new cut
```

```
%
% min z
% s.t. (xij+xkl-1)Uijkl <= z,
%

% Approximation: Approximate the mixed-integer problem by the LP [0,1] +
% rounding up in each period by Hungarian method

% Problem dimension
I = size(Uijklt,1);
J = size(Uijklt,2);
L = J;
T = size(Uijklt,5);

%tic
% Create new cut
for t = 1:T-1
   for j = 1:J
      for i = 1:I
         cut_new1(i,j,t) = sum(sum(Uijklt(i,j,:,:,t))) ...
            + sum(sum(Uijklt(:,:,i,j,t))) ...
            + sum((Vijltp(i,j,:,t)));
      end
   end
end

t = T;
for j = 1:J
   for i = 1:I
      cut_new1(i,j,t) = sum(sum(Uijklt(i,j,:,:,t))) ...
         + sum(sum(Uijklt(:,:,i,j,t)));
   end
end

for t = 1:T-1
   for l = 1:L
      for i = 1:I
         cut_new1(i,l,t+1) = cut_new1(i,l,t+1) ...
            + sum(Vijltp(i,:,l,t));
      end
   end
end

cut_new2 = sum(sum(sum(sum(sum(Uijklt))))) + sum(sum(sum(sum(Vijltp))));
%toc

% Define cost function
```

```
c_master = [zeros(1,I*J*T) 1];

% Define constraints - Add new cut to the problem
A_master = [A_master; ...
   reshape(cut_new1,1,I*J*T) -1];
b_master = [b_master; ...
   cut_new2];

% Approximate Solution for Mixed-Intger Problem:
%   Linear Programming [0,1] + Hungarian Method (Round up for each period)

%   Step 1: LP - ignore z_master at this stage
%tic
options = optimset('display','off','Largescale','off','Simplex','on');
[xijt,z_temp] = linprog(c_master,[A_master;A_TR],[b_master;b_TR],Aeq,beq,...
   [zeros(1,I*J*T) -Inf],[ones(1,I*J*T) Inf],[],options);
%[xijt,z_temp] = linprog(c_master,[A_master],[b_master],Aeq,beq,...
%    [zeros(1,I*J*T) -Inf],[ones(1,I*J*T) Inf],[],options);
xijt = reshape(xijt(1:I*J*T),I,J,T);
%toc

%   Step 2: Hungarian Method (Round up for each period) - Cost = distance
%   from 1
%tic
for t = 1:T
   xijt(:,:,t) = Hungarian(1-xijt(:,:,t));
end
%toc

%   Step 3: Recalculate z_master - find z from (in)equality constraints and
%   use z that make all constraints valid

z_temp = A_master(:,1:I*J*T)*reshape(xijt,I*J*T,1) - b_master;
z_master = max(z_temp);
```

Hungarian.m

```
function [Matching,Cost] = Hungarian(Perf)
%
% [MATCHING,COST] = Hungarian_New(WEIGHTS)
%
% A function for finding a minimum edge weight matching given a MxN Edge
% weight matrix WEIGHTS using the Hungarian Algorithm.
%
% An edge weight of Inf indicates that the pair of vertices given by its
% position have no adjacent edge.
%
% MATCHING return a MxN matrix with ones in the place of the matchings and
% zeros elsewhere.
```

```
%
% COST returns the cost of the minimum matching

% Written by: Alex Melin 30 June 2006


 % Initialize Variables
 Matching = zeros(size(Perf));

% Condense the Performance Matrix by removing any unconnected vertices to
% increase the speed of the algorithm

  % Find the number in each column that are connected
    num_y = sum(~isinf(Perf),1);
  % Find the number in each row that are connected
    num_x = sum(~isinf(Perf),2);

  % Find the columns(vertices) and rows(vertices) that are isolated
    x_con = find(num_x~=0);
    y_con = find(num_y~=0);

  % Assemble Condensed Performance Matrix
    P_size = max(length(x_con),length(y_con));
    P_cond = zeros(P_size);
    P_cond(1:length(x_con),1:length(y_con)) = Perf(x_con,y_con);
    if isempty(P_cond)
      Cost = 0;
      return
    end

    % Ensure that a perfect matching exists
      % Calculate a form of the Edge Matrix
      Edge = P_cond;
      Edge(P_cond~=Inf) = 0;
      % Find the deficiency(CNUM) in the Edge Matrix
      cnum = min_line_cover(Edge);

      % Project additional vertices and edges so that a perfect matching
      % exists
      Pmax = max(max(P_cond(P_cond~=Inf)));
      P_size = length(P_cond)+cnum;
      P_cond = ones(P_size)*Pmax;
      P_cond(1:length(x_con),1:length(y_con)) = Perf(x_con,y_con);

%**************************************************
% MAIN PROGRAM: CONTROLS WHICH STEP IS EXECUTED
%**************************************************
```

```
 exit_flag = 1;
 stepnum = 1;
 while exit_flag
   switch stepnum
     case 1
       [P_cond,stepnum] = step1(P_cond);
     case 2
       [r_cov,c_cov,M,stepnum] = step2(P_cond);
     case 3
       [c_cov,stepnum] = step3(M,P_size);
     case 4
       [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(P_cond,r_cov,c_cov,M);
     case 5
       [M,r_cov,c_cov,stepnum] = step5(M,Z_r,Z_c,r_cov,c_cov);
     case 6
       [P_cond,stepnum] = step6(P_cond,r_cov,c_cov);
     case 7
       exit_flag = 0;
   end
 end

% Remove all the virtual satellites and targets and uncondense the
% Matching to the size of the original performance matrix.
Matching(x_con,y_con) = M(1:length(x_con),1:length(y_con));
Cost = sum(sum(Perf(Matching==1)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   STEP 1: Find the smallest number of zeros in each row
%           and subtract that minimum from its row
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [P_cond,stepnum] = step1(P_cond)

 P_size = length(P_cond);

 % Loop throught each row
 for ii = 1:P_size
   rmin = min(P_cond(ii,:));
   P_cond(ii,:) = P_cond(ii,:)-rmin;
 end

 stepnum = 2;

%**************************************************************************
*******
```

```
%   STEP 2: Find a zero in P_cond. If there are no starred zeros in its
%           column or row start the zero. Repeat for each zero
%*************************************************************************
*******

function [r_cov,c_cov,M,stepnum] = step2(P_cond)

% Define variables
 P_size = length(P_cond);
 r_cov = zeros(P_size,1);  % A vector that shows if a row is covered
 c_cov = zeros(P_size,1);  % A vector that shows if a column is covered
 M = zeros(P_size);        % A mask that shows if a position is starred or primed

 for ii = 1:P_size
  for jj = 1:P_size
   if P_cond(ii,jj) == 0 && r_cov(ii) == 0 && c_cov(jj) == 0
    M(ii,jj) = 1;
    r_cov(ii) = 1;
    c_cov(jj) = 1;
   end
  end
 end

% Re-initialize the cover vectors
 r_cov = zeros(P_size,1);  % A vector that shows if a row is covered
 c_cov = zeros(P_size,1);  % A vector that shows if a column is covered
 stepnum = 3;

%*************************************************************************
*******
%   STEP 3: Cover each column with a starred zero. If all the columns are
%           covered then the matching is maximum
%*************************************************************************
*******

function [c_cov,stepnum] = step3(M,P_size)

 c_cov = sum(M,1);
 if sum(c_cov) == P_size
  stepnum = 7;
 else
  stepnum = 4;
 end

%*************************************************************************
*******
%   STEP 4: Find a noncovered zero and prime it.  If there is no starred
```

```
%         zero in the row containing this primed zero, Go to Step 5.
%         Otherwise, cover this row and uncover the column containing
%         the starred zero. Continue in this manner until there are no
%         uncovered zeros left. Save the smallest uncovered value and
%         Go to Step 6.
%*************************************************************************
*******
function [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(P_cond,r_cov,c_cov,M)

P_size = length(P_cond);

zflag = 1;
while zflag
   % Find the first uncovered zero
    row = 0; col = 0; exit_flag = 1;
    ii = 1; jj = 1;
    while exit_flag
       if P_cond(ii,jj) == 0 && r_cov(ii) == 0 && c_cov(jj) == 0
         row = ii;
         col = jj;
         exit_flag = 0;
       end
       jj = jj + 1;
       if jj > P_size; jj = 1; ii = ii+1; end
       if ii > P_size; exit_flag = 0; end
    end

   % If there are no uncovered zeros go to step 6
    if row == 0
      stepnum = 6;
      zflag = 0;
      Z_r = 0;
      Z_c = 0;
    else
      % Prime the uncovered zero
      M(row,col) = 2;
      % If there is a starred zero in that row
      % Cover the row and uncover the column containing the zero
       if sum(find(M(row,:)==1)) ~= 0
         r_cov(row) = 1;
         zcol = find(M(row,:)==1);
         c_cov(zcol) = 0;
       else
         stepnum = 5;
         zflag = 0;
         Z_r = row;
         Z_c = col;
```

```
        end
    end
end

%*************************************************************************
*******
% STEP 5: Construct a series of alternating primed and starred zeros as
%       follows.  Let Z0 represent the uncovered primed zero found in Step 4.
%       Let Z1 denote the starred zero in the column of Z0 (if any).
%       Let Z2 denote the primed zero in the row of Z1 (there will always
%       be one).  Continue until the series terminates at a primed zero
%       that has no starred zero in its column.  Unstar each starred
%       zero of the series, star each primed zero of the series, erase
%       all primes and uncover every line in the matrix.  Return to Step 3.
%*************************************************************************
*******

function [M,r_cov,c_cov,stepnum] = step5(M,Z_r,Z_c,r_cov,c_cov)

 zflag = 1;
 ii = 1;
 while zflag
  % Find the index number of the starred zero in the column
  rindex = find(M(:,Z_c(ii))==1);
  if rindex > 0
   % Save the starred zero
   ii = ii+1;
   % Save the row of the starred zero
   Z_r(ii,1) = rindex;
   % The column of the starred zero is the same as the column of the
   % primed zero
   Z_c(ii,1) = Z_c(ii-1);
  else
   zflag = 0;
  end

  % Continue if there is a starred zero in the column of the primed zero
  if zflag == 1;
   % Find the column of the primed zero in the last starred zeros row
   cindex = find(M(Z_r(ii),:)==2);
   ii = ii+1;
   Z_r(ii,1) = Z_r(ii-1);
   Z_c(ii,1) = cindex;
  end
 end

 % UNSTAR all the starred zeros in the path and STAR all primed zeros
```

```
 for ii = 1:length(Z_r)
  if M(Z_r(ii),Z_c(ii)) == 1
   M(Z_r(ii),Z_c(ii)) = 0;
  else
   M(Z_r(ii),Z_c(ii)) = 1;
  end
 end

 % Clear the covers
 r_cov = r_cov.*0;
 c_cov = c_cov.*0;

 % Remove all the primes
 M(M==2) = 0;

stepnum = 3;

%
**********************************************************************
****
% STEP 6: Add the minimum uncovered value to every element of each covered
%        row, and subtract it from every element of each uncovered column.
%        Return to Step 4 without altering any stars, primes, or covered lines.
%**********************************************************************
*******

function [P_cond,stepnum] = step6(P_cond,r_cov,c_cov)
a = find(r_cov == 0);
b = find(c_cov == 0);
minval = min(min(P_cond(a,b)));

P_cond(find(r_cov == 1),:) = P_cond(find(r_cov == 1),:) + minval;
P_cond(:,find(c_cov == 0)) = P_cond(:,find(c_cov == 0)) - minval;

stepnum = 4;

function cnum = min_line_cover(Edge)

 % Step 2
  [r_cov,c_cov,M,stepnum] = step2(Edge);
 % Step 3
  [c_cov,stepnum] = step3(M,length(Edge));
 % Step 4
  [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(Edge,r_cov,c_cov,M);
 % Calculate the deficiency
  cnum = length(Edge)-sum(r_cov)-sum(c_cov);
```

dp.m

```
function [TC_opt,xijt_opt,xijt0] = dp(Cijklt,Rijltp,I,J,T,xijt0,xijt1)
%
%   Dynamic Programming
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Dynamic programming for dynamic quadratic assignment problem

% Input - given xijt (layout)

if nargin < 6   %   no xijt input
   % Construct layout from permutation
   loind = perms(1:I);
   LO = size(loind,1);
   for ilo = 1:LO
      xijt0(:,:,ilo) = zeros(I);
      for i = 1:I
         xijt0(i,loind(ilo,i),ilo) = 1;
      end
   end
else           %   xijt input
   % Combine layout in every period into one big layout used for every period
   %   and remove repeating layout

   Ncut = size(xijt0,4);
   LO = T*Ncut; % total number of layout (Ncut x T)

   xijt0 = reshape(xijt0,1,I*J,LO);
   xijt0 = squeeze(xijt0)';

   xijt0 = unique(xijt0,'rows');
   LO = size(xijt0,1);
   xijt0 = reshape(xijt0',I,J,LO);

   % Add layout from previous iterations

   if size(xijt1,1) == I
      Ncut = size(xijt1,3);
      for ilo = LO+1:LO+Ncut
         xijt0(:,:,ilo) = xijt1(:,:,ilo-LO);
      end
      LO = Ncut + LO;

      % Remove repeating layout
      xijt0 = reshape(xijt0,1,I*J,LO);
```

```
        xijt0 = squeeze(xijt0)';

        xijt0 = unique(xijt0,'rows');
        LO = size(xijt0,1);
        xijt0 = reshape(xijt0',I,J,LO);
    end
end

% Copy all the unique to every period and compute assignment cost over
%   different layout
for ilo = 1:LO
    for t = 1:T
        xijt(:,:,t,ilo) = xijt0(:,:,ilo);
        AC(ilo,t) = AssignmentC(Cijklt(:,:,:,:,t),xijt(:,:,t,ilo));
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Sort Layout by assignment cost
[ACmin,Imin]=sort(AC);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate lower bound
LB = sum(ACmin(1,:));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate upper bound

UB = zeros(1,T+1);

% Upper bound 1 - LB + Rearrangement Costif
%   Rearrangement cost
RC = zeros(1,T-1);
for t = 1:T-1
    RC(t) = RearrangementC(Rijltp(:,:,:,t),xijt(:,:,t,Imin(1,t)),...
        xijt(:,:,t+1,Imin(1,t+1)));
end
UB(1) = LB + sum(RC);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Upper bound 2 to T+1 - Same assignment; No rearrangment - same layout

for t = 1:T
```

```
   UB(t+1) = sum(AC(Imin(1,t),:));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate different bound
DB = min(UB) - LB;

% Have we found optimal layout i.e. DB = 0
if DB == 0
   TC_opt = LB;
   for t = 1:T
      xijt_opt = xijt(:,:,:,Imin(1,t));
   end
   return
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check diff. cost v.s. DB
DC = ACmin - ones(LO,T)*diag(ACmin(1,:));   % diff cost for each period

% Find the number of layout in each period that has AC < DB;
for t = 1:T;
   nlayout_dp(t)= nnz(min(DC(:,t)-DB,0));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate all combination
Ncomb = prod(nlayout_dp);
Nblock = Ncomb; % number of total blocks for each layout in a given period

for t = 1:T
   index = 1;
   Nblock = Nblock/nlayout_dp(t);
   while (index <= Ncomb)
      for i = 1:nlayout_dp(t);
         for l = 1:Nblock
            layout_dp(index,t) = i;
            index = index + 1;
         end
      end
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Add rearrangement cost to all possible candidates
TC = zeros(Ncomb,1);

for index = 1:Ncomb
   % pull layout and its cost
   layout_i = layout_dp(index,:);
   for t = 1:T
      xijt_i(:,:,t) = xijt(:,:,t,Imin(layout_i(:,t),t));
      AC_i(t) = ACmin(layout_i(:,t),t);
   end

   % add rearrangement cost
   for t = 1:T-1;
      RC_i(t) = RearrangementC(Rijltp(:,:,:,t),xijt_i(:,:,t),...
         xijt_i(:,:,t+1));
   end
   TC(index) = sum(AC_i) + sum(RC_i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Find optimal layout

[TC_opt,index_opt]=min(TC);
layout_i = layout_dp(index_opt,:);
for t = 1:T
   xijt_opt(:,:,t) = xijt(:,:,t,Imin(layout_i(:,t),t));
end
```

APPENDIX D

ECLiPSE Source Code

ECLiPSe Source code for the Logic-Based model of DQAP

```
:- lib(fd).
% :- lib(ic_global).
% :- lib(fdplex).
:- lib(branch_and_bound).
:- lib(array_constraints).

main( N, P, Cost, Y):-

% dim( F, [P, N, N]),
% dim( D, [P, N, N]),

F = []([](
        [](0,36,6,66,12,83,13,58,66,80,60,45,76,80,27,14,11,47,95,47,93,80,11,39,87,
23,46,99,42,26,72,36,62,0,80,24,14,48,24,49),


%% line 130
D = []([](
        [](0,10,35,32,4,54,15,44,49,37,79,24,21,92,95,97,26,37,5,24,92,2,74,0,45,18,3
9,78,42,97,74,55,6,69,56,5,77,69,41,78),


%% line 251
R =
[]([](0,38,74,89,71,39,66,42,78,87,94,5,3,55,43,65,45,16,47,73,97,15,25,49,4,56,58,8
9,53,48,22,68,14,90,56,48,89,43,51,30),


%% line 292
dim( Y, [P, N]),
% dim( X, [P, N]),  %% temp variables for rearranging L/O

%% collect the variables in Y
(for (I, 1, P), fromto([], Yv_in, Yv_out, Y_varlist),
 param(Y, J, N, P)
 do
        (for(J, 1, N), fromto( Yv_in, Yv_in2, Yv_out2, Yv_out),
         fromto( [], Lv_in, Lv_out, L_varlist),
         param(Y, I, P, N)
         do
                Yij is Y[I,J],    %%  element lookup
                % Xij is X[I,J],
                Yv_out2 = [Yij| Yv_in2],
                Lv_out = [Yij| Lv_in]
        ),
```

```
        L_varlist :: 1..N,              %% layout of each period
        alldifferent(L_varlist)
),

% X :: 1..N,
Y_varlist :: 1..N,
% alldifferent(Y_varlist),

(for(T, 1, P), fromto (0, In1, Out2, Objective),
 param (N, F, D, R, Y)
 do
        (for(I, 1, N), fromto(In1, In2, Out3, Out2),
         param(N, F, D, R, Y, T)
         do
                (for(K, 1, N), fromto(In2, In3, C1+In3, Out3),
                 param(N, F, D, R, Y, T, I)
                 do
                        Yit is Y[T,I],
                        Ykt is Y[T,K],
                        Fikt is F[T,I,K],
                        writeln({Dyt = [T, Yit, Ykt]}),
                            { Dyt = D[T, Yit,Ykt] },    %% array constraint!
                        C1 #= Fikt * Dyt
                )
        )
),
Cost1 #= Objective,

 P1 is P-1,
(for(T, 1, P1), fromto(0, In1, Out2, Rcost),
 param(N, R, Y, I)
 do
        (for(I, 1, N), fromto(In1, In2, C+In2, Out2),
        param(T, R, Y)
        do
                B is T+1,
                Yia is Y[T,I],
                Yib is Y[B,I],
                { Riab = R[Yia,Yib] },
                C #= Riab
        )
 ),
 Cost2 #= Rcost,
 Cost #= Cost1 + Cost2,
% labeling(Y_varlist).
min_max(labeling( Y_varlist), Cost).
% bb_min(labeling(Y_varlist), Cost, bb_options with [strategy:continue]).
```

```
% minimize(labeling(Y_varlist),Y_varlist,Y_varlist,Cost).
% bb_min(labeling(Y_varlist), Cost, bb_options with [strategy:step,timeout:10800]).
```

CURRICULUM VITAE

NAME : Miss Sirirat Muenvanichakul

BIRTH DATE : June 11, 1970

BIRTH PLACE : Bangkok, Thailand

EDUCATION :

YEAR	INSTITUTION	DEGREE/DIPLOMA
1993	Kasetsart Univ.	B.Eng. (Industrial Engineering)
1998	Kasetsart Univ.	M.Eng. (Industrial Engineering)

SCHOLARSHIP : Faculty of Engineering at Si Racha, Kasetsart University, Si Racha Campus