Original Research Paper

# Applications of Parallel Computing for Facility Location-Transportation Problems for Disaster Response

[1]Chansiri Singhtaun and [2]Suriya Natsupakpong

[1]*Department of Industrial Engineering, Kasetsart University, Bangkok, Thailand*
[2]*Institute of Field Robotics, King Mongkut's University of Technology Thonburi, Bangkok, Thailand*

*Article history*
Received: 16-03-2015
Revised: 04-05-2015
Accepted: 26-05-2015

Corresponding Author:
Chansiri Singhtaun
Department of Industrial Engineering, Kasetsart University, Bangkok, Thailand
Email: chansiri.s@ku.ac.th

**Abstract:** This study extends the capability and increases the performance of a traditional Branch and Bound (BB) algorithm to solve Facility Location-Transportation problems for Disaster Response (FLTDR) using parallel computing. A Two-Stage Branch and Bound (TBB) algorithm was developed to support two parallel computing approaches. This algorithm divides problems into two small sub-problems, which are a facility location sub-problem and a transportation sub-problem. All possible numbers of distribution centers are determined. All possible locations relating to any number of distribution centers are explicitly explored. The transportation sub-problem corresponding to any selected location is then solved. Two parallel approaches for the TBB algorithm (PTBB) differ in partitioning the list of sub-problems. The first approach (PTBB1) solves both sub-problems in parallel. The other (PTBB2) explores the locations in sequence and solves only the transportation sub-problems in parallel. The numerical experiments were conducted on various sizes of generated problems. The quality of the solution and the computing time of both approaches were compared with a BB algorithm with premature termination by time. The experimental experiences showed that both PTBB1 and PTBB2 are more efficient and effective than a BB algorithm. However, the PTBB1 should be suggested for the FLTDR because of the least computational time usage.

**Keywords:** Branch-and-Bound Algorithms, Disaster Response, Facility Location, Parallel Computing, Transportation

## Introduction

The number of natural disasters has increased and the severity has grown over recent years. During 2007-2011, natural disasters killed 23.64% more people and there was a 59.21% increase in economic damage than during 2002-2006 (UNISDR, 2012). Efficient and effective disaster operation management has become a vital research topic. The life-cycle of disaster operations management comprises four phases, which are the mitigation phase, the preparedness phase, the response phase and the recovery phase (Altay and Green, 2006). The first two phases are pre-positioning phases that need to be performed prior to the onset of a disaster. The other two are post-disaster phases. The period of time in each phase depends on the type of disaster (a quick-onset or a slow-onset disaster). The disaster response is a crucial phase. The objective of disaster response in the humanitarian relief chain is to rapidly provide relief (emergency food, water, medicine, shelter and supplies) to areas affected by large-scale emergencies, so as to minimize human suffering and death (Balcik and Beamon, 2008). Most research topics have emphasized designing a disaster management framework, such as the study appearing in Aslanzadeh *et al*. (2009). Few research papers have focused on constructing a disaster response operation framework and application. The research in the latter area aims to determine a solution by using a mathematical method. It starts with identifying the problems and formulating the mathematical model to represent the real problems. Next, an efficient method has to be found to give the best or at least a good quality solution for the mathematical model. Finally, the solution to answer the real problems is interpreted and validated. Relief logistics play an important role in this framework. The scope of relief logistics relates to ten subsystems, which are planning, inventory distribution, transportation, procurement, maintenance, control, human resources, information and communication and administration subsystems (Aslanzadeh *et al*., 2009). The first three subsystems have been intensively studied under the following topics: facility location problems, inventory problems, transportation/ routing problems and scheduling problems. Both individual analyses and the integration of these four problems have been researched. The FLTDR is an emergency logistics aspect of the response phase. The framework for the main emergency logistics activities and their associated facilities and flows were proposed by Caunhye *et al*. (2012) and are summarized in Fig.1.

The arrows in Fig. 1 indicate the activities and the directions of the main flows of activity. Evacuation deals with the flow of people, relief distribution with resources and casualty transportation of wounded people. Non-directional arrows do not indicate flows but rather express that a relationship exists between two components.

The FLTDR relates to solving both location and transportation problems simultaneously. The location problem requires designing a network for distributing humanitarian aid (e.g., water, food, medical goods and survival equipment). It mainly consists of determining the number, the position and the mission of a humanitarian aid distribution center within the disaster region. The transportation problem deals with the distribution of humanitarian aid from the distribution center to demand points (Abounacer *et al*., 2014). Most of the mathematical models for FLTDR are Mixed Integer linear Programming (MIP) problems with complex constraint structures (Minoux, 1989). The BB algorithm is currently the only general tool available for finding optimal solutions to these difficult formulations (Bourbeau *et al*., 2000). However, finding an optimal solution for a complex and large size FLTDR using a BB takes excessive computing time. Parallel computing is one of the most efficient alternatives that have been used since the beginning of the twenty-first century. The use of parallelism to speed up the execution of a typical (sequential) BB algorithm is widely known as a Parallel Branch and Bound (PBB) algorithm.

In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem (Bader *et al*., 2005). However, the choice of computer architectures strongly influences the design structure and performance of a parallel computation. The distinctions in the hardware level can be classified by five parameters (Gendron and Crainic, 1994), which are the control parameter, synchronization, the grain, the communication parameter and the number of processors. Control-flow parallel architectures with only one control unit belong to the Single Instruction Multiple Data (SIMD) class, while systems with several control units (generally one per processor) belong to the MIMD (multiple instruction multiple data) class. When there is only one clock, we speak of a synchronous system, while in the presence of several clocks, typically one per processor, the system is called asynchronous. In fine-grained systems, each processor can handle only a small amount of data, corresponding to scalar or small vector operations. At the other extreme, coarse-grained systems are characterized by the possibility of simultaneous treatment of large amounts of data. When processors write and read instructions in a common memory, the systems are called shared-memory systems. When processors only exchange messages, systems are called message-passing systems. Last, massively parallel systems are made of a large number of processors, in the order of thousands. However, by adding appropriate software mechanisms, it is possible with some systems to simulate the behavior of other systems.

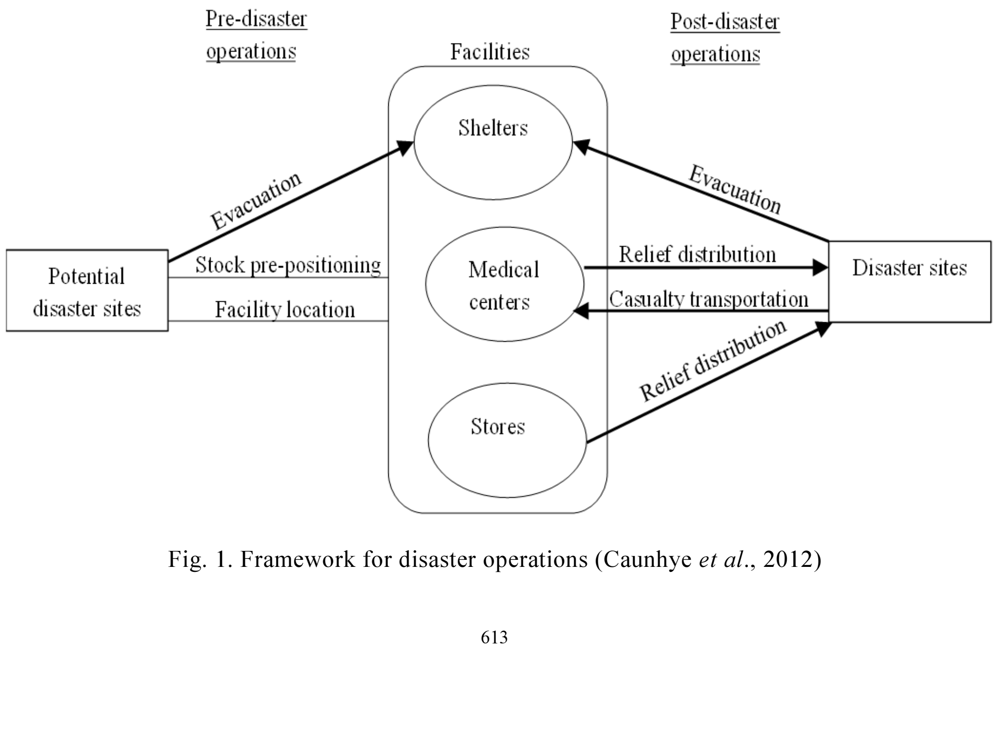

Fig. 1. Framework for disaster operations (Caunhye *et al*., 2012)

There is a wide range of software for BB algorithms because the numbers of problem types, user interface types, parallelism types and machine types are large. Software for which there exists an interface to implement a typical BB algorithm, which means that one can use this type of library to implement one's own bounding procedure and one's branching procedure, are summarized in Crainic *et al*. (2006). Libraries or applications exist that are dedicated to one type of problem; commercial linear solvers like CPLEX, Xpress-MP, MATLAB or Open Source Software like GLPK or lp_Solve are dedicated to solve MIP problems but the BB part is hidden. It cannot be customized to implement an efficient parallel one. However, these solvers could be used using a callable library. An application that implements a PBB algorithm could use this kind of solver to compute the bound (Crainic *et al*., 2006). There are three main approaches of PBB algorithms according to the degree of parallelism of the search tree. Parallelism of type 1 introduces parallelism when performing the operations on generated sub-problems (e.g., bounding computations). In the type 2 approach, the search tree is built in parallel by performing operations on several sub-problems simultaneously. In parallelism of type 3, several trees are explored concurrently (Gendron and Crainic, 1994). Since both selection of computer architectures and PBB approaches for a particular MIP problem affect the computational performance, most of the researches in the PBB approach area usually emphasize developing new or improving the existing computer architectures and PBB algorithm approaches. Many researches are dedicated to comparisons of applying various architectures or PBB algorithm approaches in order to suggest the appropriate algorithm for an on-hand MIP problem. For example, Barreto and Bauer (2010) compared efficiency of PBB implementation between MPI (distributed memory parallel model) and OpenMP (shared memory parallel model) implementations on a computer cluster connecting through network. The results showed that the MPI took the least time in all executions. However, a better way to keep all processors working full time was needed for these implementations. The efficient load balancing strategies proposed by Otten and Dechter (2010) had a crucial role to solve this problem. Recently, PBB algorithm was popularly implemented on a multi-core CPU system (Leroy *et al*., 2014), a multi-thread GPU system (Chakroun *et al*., 2013) or a hybrid CPU-GPU system (Boukedjar *et al*., 2012). However, the strategies to minimize the time spent on communication between cores or threads executing the PBB algorithm were needed to improve the systems.

In this study, we study the applications of parallel computing with our proposed algorithm for FLTDR (a specific MIP problem) using an available function called "*parfor*" in MATLAB. The proposed algorithm is a BB based algorithm. The goal is to study the efficiency and effectiveness of two parallel approaches using the efficient available commercial software to improve the computational performance of a typical BB algorithm. In order to do that, we separate the problem into small sub-problems instead of using the classical PBB approaches that try to solve a large MIP problem in one time in a parallel manner. The remaining sections of this paper are as follows. The next section we explain the FLTDR problem and the mathematical model. After that, we propose the parallel approaches, identify the compute platform implementing parallel computing and explain the data generating method for numerical experiments. The results are discussed in the following section. Finally, the conclusions are drawn from the results to suggest an appropriate parallel approach for FLTDR.

## Mathematical Model Formulation

### *Problem Description*

The FLTDR in this study focuses on calculating the number of distribution centers to be constructed; determining the locations of distribution centers; identifying the quantity of relief items to be stored and determining the assignment of vehicles to supply the humanitarian aid items so as to maximize the relief item coverage under the following assumptions. Each particular house or building within the affected area could require humanitarian aid and is thus a potential demand point. The demand quantities are estimated by a homeland security organization or experts. The demand quantities can only be satisfied by the distribution center, which is assumed to stock and distribute multiple types of relief item. The relief items are divided with respect to their response time criticalities and target response time intervals.

The amount of stock to be held at the distribution center depends on the number and location of distribution centers in the network as well as the assignment of demand locations to the distribution centers, while distribution center location and assignment decisions are affected by the quantity of relief items to be stocked at each distribution center. Each distribution candidate site has a global and a per product capacity that fixes the maximum quantity to be stored within the site. The location candidates and the capacity of distribution centers are considered in the pre-disaster phase based on the demand locations and quantities. Both location and stock decisions are limited by pre-disaster budgetary restrictions.

The vehicles available at candidate sites are of various types and there are different numbers of available vehicles. The different docking times of each vehicle type at each site and the time needed for loading and unloading one unit of each product for each vehicle type are considered. The traveling time from a distribution center to a demand location is determined corresponding to distance and vehicle type. There are also some restrictions on the total weight and the total volume of vehicles. A maximum daily work time for each vehicle type is imposed. A given vehicle can perform as many trips as needed during a day as long as the corresponding work time limit is respected. Each vehicle trip is assumed to visit only one demand point at a time. One demand point may be visited many times. However, because of the maximum daily work time, the number of trips to a specific delivery point by a particular vehicle will be limited to a maximum value, which is set at two. Finally, shipping costs from distribution centers to demand points are restricted by post-disaster budgetary restrictions. Next, the mathematical model formulation of this problem is presented.

### *Parameter Description*

The proposed mathematical model was modified from Abounacer *et al.* (2014) by creating a new objective function, adding budgetary constraints and changing the transportation cost function to make the problem match the real situation. The parameters and decision variables are defined as follows:

| | |
|---|---|
| $I$ | Set of demand points; $I = \{1,\ldots, n\}$ |
| $J$ | Set of items; $J = \{1,\ldots, p\}$ |
| $L$ | Set of candidate sites; $L = \{1,\ldots, u\}$ |
| $H$ | Set of vehicle types at site $l$; $H = \{1,\ldots, m_l\}$ |
| $K$ | Set of number of vehicles for each vehicle type at site $l$; $K = \{1,\ldots, u_{hl}\}$ |
| $V$ | Set of vehicle trip; $V = \{1,2\}$ |
| $d_{ij}$ | Demand for item type $j$ at demand point $i$ |
| $s_{jl}$ | Capacity of site $l$ for item type $j$ |
| $S_l$ | Capacity of site $l$ for all item |
| $Q_h$ | Weight capacity of a vehicle of type $h$ |
| $V_h$ | Volume capacity of a vehicle of type $h$ |
| $\tau_{lh}$ | Docking time for a vehicle of type $h$ at site $l$ |
| $t_{ilh}$ | Travel time from site $l$ to demand point $i$ by vehicle type $h$ |
| $\alpha_{jh}$ | Time of loading and unloading one unit of item type $j$ into a vehicle of type $h$ |
| $D_h$ | Maximum daily work time for a vehicle of type $h$ |
| $w_j$ | Weight of one unit of item type $j$ |
| $v_j$ | Volume of one unit of item type $j$ |
| $F_l$ | Fixed cost of establishing distribution center $l$ |
| $g_{jl}$ | Unit cost of acquiring and storing item type $j$ at distribution center $l$ |
| $c_{ilh}$ | Unit cost of shipping items from distribution center $l$ to demand point $i$ by vehicle type $h$ |
| $B_0$ | Emergency relief budgets allocated for pre-positioning relief supplies |
| $B_1$ | Emergency relief budgets allocated for post-disaster distribution |

### *Decision Variables*

$$y_l = \begin{cases} 1 & \text{if a distribution center is located at site } l \\ 0 & \text{otherwise} \end{cases}$$

$$X_{ilhkv} = \begin{cases} 1 & \text{If demand point } i \text{ is visited from distribution center } l \text{ with the } k^{th} \text{vehicle of type } h \text{ on its } v^{th} \text{trip} \\ 0 & \text{otherwise} \end{cases}$$

$Q_{ijlhkv}$ = Quantity of item type $j$ delivered to point $i$ from distribution center $l$ with the $k^{th}$vehicle of type $h$ on its $v^{th}$ trip

$p_{jl}$ = Quantity of item type $j$ provided at site $l$

### *Mathematical Model*:

$$Max\ Z = \sum_{i\in I}\sum_{j\in J}\frac{1}{d_{ij}}\sum_{l\in L}\sum_{h\in H}\sum_{k\in K}\sum_{v\in V}Q_{ijlhkv} \tag{1}$$

Subject to:

$$\sum_{l\in L}\sum_{h\in H}\sum_{k\in K}\sum_{v\in V}Q_{ijlhkv} \le d_{ij} \quad \forall i\in I, j\in J \tag{2}$$

$$\sum_{i\in I}\sum_{h\in H}\sum_{k\in K}\sum_{v\in V}Q_{ijlhkv} \le p_{jl} \quad \forall j\in J, l\in L \tag{3}$$

$$\sum_{i\in I}\sum_{k\in K}\sum_{v\in V}\left(\left(2t_{ilh}+\tau_{lh}\right)X_{ilhkv}+\sum_{j\in J}\alpha_{jh}Q_{ijlhkv}\right) \le D_h y_l \quad \forall h\in H, l\in L \tag{4}$$

$$\sum_{j\in J}w_j Q_{ijlhkv} \le Q_h X_{ilhkv} \quad \forall i\in I, l\in L, h\in H, k\in K, v\in V \tag{5}$$

$$\sum_{j\in J}v_j Q_{ijlhkv} \le V_h X_{ilhkv} \quad \forall i\in I, l\in L, h\in H, k\in K, v\in V \tag{6}$$

$$\sum_{j\in J}p_{jl} \le S_l y_l \quad \forall l\in L \tag{7}$$

$$p_{jl} \le s_{jl} \quad \forall j\in J, l\in L \tag{8}$$

$$\sum_{l\in L}F_l y_l + \sum_{j\in J}\sum_{l\in L}g_{jl}p_{jl} \le B_0 \tag{9}$$

$$\sum_{i\in I}\sum_{l\in L}\sum_{h\in H}\sum_{k\in K}\sum_{v\in V}X_{ilhkv}c_{ilh} \le B_1 \tag{10}$$

$$y_l \in \{0,1\} \tag{11}$$

$$X_{ilhkv} \in \{0,1\} \tag{12}$$

$$Q_{ijlhkv} \ge 0, integer \tag{13}$$

$$p_{jl} \geq 0, integer \tag{14}$$

The objective function Equation 1 maximizes the total fraction of demand covered by the established distribution centers. Constraint set Equation 2 guarantees that the quantity of item *j* delivered for each demand point *i* does not exceed its demand. Constraint set Equation 3 ensures that the total quantity of a given item type *j* delivered from a distribution center *l* does not exceed the quantity of item type *j* available in this distribution center. Constraint set Equation 4 requires that the maximum daily work time restriction related to each vehicle *k* of type *h* located at a distribution center *l* is not exceeded. These constraints also prohibit trips from unopened sites. Constraint sets Equations 5 and 6 express the vehicle capacity constraints for each trip in terms of weight and volume. Constraint set Equations 7 and 8, respectively, insure that the total and the per item capacity of the distribution center are satisfied. Constraint Equation 9 requires that the pre-disaster expenditure related to establishing a distribution center and holding inventory does not exceed the pre-disaster budget. Constraint Equation 10 ensures that the total transportation costs do not exceed the post-disaster budget. Finally, constraint sets Equations 11-14 define the nature of decision variables used in the model.

## Research Methodology

In this section, the proposed exact algorithm, which is called the TBB algorithm, for FLTDR is described. The two parallel approaches applied with the algorithm as well as the method to construct the test problems are also clearly explained.

### *The Two-Stage Branch and Bound Algorithm*

The TBB algorithm splits the complex and large size problems into two small sub-problems: The facility location problem and the transportation problem. The iterative process is then carried out as follows:

Step 1: Calculate the upper bound of the number of distribution centers to be located ($ub_{NumDC}$) using the budgetary constraint as follows:

$$ub_{NumDC} = \max\left\{1, \left\lceil \frac{B_0}{\sum_{l\in L} F_l} \right\rceil \right\} \tag{15}$$

Let the set of current solutions ($y_l$, $X_{ilhkv}$ and $Q_{ilhkv}$) be an empty set and the current objective function ($Z_{cur}$) is zero.

Step 2: Set the current number of distribution centers to be located ($NumDC_{cur}$) at 1.

Step 3: Find all possible patterns of selecting $NumDC_{cur}$ locations out of *u* candidate locations. Now all possible sets of decision variables $y_l$ corresponding to $NumDC_{cur}$ are created. Let the number of all possible patterns corresponding to $NumDC_{cur}$ be $NumPat_{cur}$.

Step 4: Set the current pattern ($Pat_{cur}$) at 1.

Step 5: Select the set of decision variables $y_l$ relating to $NumDC_{cur}$ and $Pat_{cur}$.

Step 6: Solve a transportation sub-problem relating to $y_l$ using a BB algorithm. At this step the solutions for variables $X_{ilhkv}$ and $Q_{ilhkv}$ are found and the objective function *(Z)* corresponding to $y_l$ is known.

Step 7: Update the set of current solutions and $Z_{cur}$ by employing a new solution and a new *Z* obtained from step 6 if the *Z* is better (more) than $Z_{cur}$. Otherwise, go to step 8.

Step 8: Set $Pat_{cur} = Pat_{cur}+1$. If $Pat_{cur} \leq NumPat_{cur}$ go to step 9. Otherwise, go to step 10.

Step 9: Select the set of decision variables $y_l$ relating to a new $Pat_{cur}$. Solve the LP relaxation problem of the transportation sub-problem using an interior point algorithm. If $Z > Z_{cur}$, go to step 6. Otherwise, go to step 8.

Step 10: Set $NumDC_{cur} = NumDC_{cur}+1$. If $NumDC_{cur} \leq ub_{NumDC}$ go to step 3. Otherwise, stop the iterative process.

### *Parallel Approaches*

We propose two parallel computing approaches, which are different in the job distribution method, in order to study the effect of job assignment on computing time. These two programs apply the same command function, "*parfor*" function in MATLAB, to do parallel computing at a different part of the algorithm. The first approach (PTBB1), applies the function with step 2 to step 10. The problems that have different values of $NumDC_{cur}$ are delivered to the six parallel workers (cores) in a parallel manner. Once the problem with a specific $NumDC_{cur}$ is assigned to a worker, all possible patterns of selecting the locations corresponding to that $NumDC_{cur}$ are solved at that worker. After solving all the problems of all patterns, the next assigned problem relating to a next specific $NumDC_{cur}$ is then solved. The second approach (PTBB2) applies the function with step 4 to step 9. The problems are selected in the sequence of $NumDC_{cur}$. Once $NumDC_{cur}$ is fixed, the problems relating to all possible patterns of that $NumDC_{cur}$ are solved in a parallel manner. The process moves to the next $NumDC_{cur}$, after these problems are already solved.

### *Numerical Experiments*

In order to measure the performance of parallel computing, the results of PTBB1 and PTBB2 are compared with the results of a sequential BB algorithm. These three algorithms are coded with MATLAB. The numerical experiments are implemented on an asynchronous shared memory system, which is constructed from a workstation with a CPU Intel Core i7-5820K 3.30 GHz 6-core processor with 16 GB RAM.

Sixteen problem cases are created according to the number of demand points ($n$) and the number of candidate locations of distribution centers ($u$). The various sizes of $n$ are 5, 10, 15 and 20. The various sizes of $u$ are 1, 2, 3 and 4. The data sets of sixteen problem cases are generated as follows. The number of item types ($p$) is fixed at 2. The number of vehicle types at each site ($m_l$) and the number of vehicles for each vehicle type at site $l$ are randomized between 1 and 5. The parameters $d_{ij}$, $p_{jl}$ and the parameters existing on the left of the constraint sets Equations 4 to 10 are obtained by random generation. The others are randomly selected from the determined range to avoid the number of feasible solutions being limited to a small number. The minimum and maximum values of the range are fixed by respectively multiplying 0.5 and 300 with the value on the left hand side. Three instances are generated for each data set. The BB algorithm used to solve these instances are set to be prematurely terminated at 28,800 sec or 8 h in order to limit the computational time for large-size problems. However, computational time limitations are not set for PTBB1 and PTBB2. To solve FLTDR with a traditional BB algorithm and to solve a transportation sub-problem, which is an MIP problem, in step 6 of the TBB algorithm, the command "*intlinprog*" is used. Some optimization options are specifically set as follows in order that the algorithms are stopped because of only time limitation. The maximum number of nodes is set at 1e50. The difference between the internally calculated upper and lower bounds on the objective function is set at 1e-30. The other options are set at default. All options for solving an LP relaxation problem in step 8 are set at the default. The percentage of demand coverage, computing time and the solutions of decision variables are recorded.

## Results and Discussion

The average computing time and percentage of demand coverage (the objective function value in terms of percentage) for all cases of all algorithms are shown in Table 1. To express the trend of computing time clearly, Fig. 2 is created.

From Table 1, the computing time of three approaches (BB, PTBB1 and PTBB2 approach) increase according to both $n$ and $u$. The sequential BB algorithm application is limited for small size problems. Only for problem sizes of $n = 5$, $u \le 3$ and $5 < n \le 30$, $u = 1$ can the algorithm give the solution within 8 h. Therefore, solving problem sizes of $n = 5$, $u = 4$ and $n > 5$, $u > 1$ using this algorithm may not provide the optimal solution because of premature termination. However, the BB algorithm gives a solution gap within 0.75% inferior than the optimal solution. Unlike the BB algorithm, both parallel approaches, PTBB1 and PTBB2, can give optimal solutions for all problem cases within 8.01 h.

Table 1. Average computing time and average percentage of demand coverage of BB, PTBB1 and PTBB2 approach

| | | | Average computing time (seconds) | | | Average percentage of demand coverage | | |
|---|---|---|---|---|---|---|---|---|
| Case | $n$ | $u$ | BB | PTBB1 | PTBB2 | BB | PTBB1 and PTBB2 | Percentage of Solution gap |
| 1 | 5 | 1 | 60.27 | 38.75 | 41.09 | 18.72 | 18.72 | 0.00 |
| 2 | 5 | 2 | 14,539.10 | 1,695.31 | 3,225.04 | 43.28 | 43.32 | -0.09 |
| 3 | 5 | 3 | 19,217.72 | 9,606.67 | 12,199.85 | 44.62 | 44.96 | -0.75 |
| 4 | 5 | 4 | 28,801.00 | 23,079.67 | 26,802.00 | 57.51 | 57.70 | -0.33 |
| 5 | 10 | 1 | 10,027.91 | 2,616.84 | 2,688.94 | 13.05 | 13.04 | 0.00 |
| 6 | 10 | 2 | 28,801.00 | 8,092.41 | 10,760.18 | 31.15 | 31.16 | -0.02 |
| 7 | 10 | 3 | 28,801.00 | 18,015.88 | 21,609.91 | 37.29 | 37.29 | 0.00 |
| 8 | 10 | 4 | 28,801.67 | 24,031.44 | 28,821.99 | 41.11 | 41.11 | 0.00 |
| 9 | 15 | 1 | 15,360.25 | 3,721.12 | 3,726.75 | 12.12 | 12.12 | 0.00 |
| 10 | 15 | 2 | 28,800.00 | 7,211.96 | 10,725.04 | 19.90 | 19.90 | 0.00 |
| 11 | 15 | 3 | 28,801.00 | 9,900.86 | 18,036.83 | 30.32 | 30.43 | -0.35 |
| 12 | 15 | 4 | 28,831.33 | 26,428.00 | 28,829.67 | 43.09 | 43.09 | 0.00 |
| 13 | 30 | 1 | 14,812.58 | 353.94 | 364.27 | 7.13 | 7.13 | 0.00 |
| 14 | 30 | 2 | 28,802.00 | 7,205.80 | 14,433.00 | 8.75 | 8.75 | 0.00 |
| 15 | 30 | 3 | 28,802.33 | 16,817.50 | 21,628.67 | 15.68 | 15.68 | 0.00 |
| 16 | 30 | 4 | 28,801.00 | 28,342.00 | 28,740.00 | 17.94 | 17.94 | 0.00 |

At the same *u*, the average percentage of demand coverage decreases when *n* increases. At the same *n*, the average percentage of demand coverage tends to be higher when *u* increases. Moreover, from the problem solutions of all cases, all candidate locations are used. These results show that to meet more demand from disaster victims, more distribution centers are needed.

According to Fig. 2, at the same *n*, the computing time of all approaches tend to have positive nonlinear relations with *u*. However, this nonlinear trend is not stronger when *n* increases. This is because problem sizes are extremely extended even when *u* increases by one. When one candidate location is added, the decision variables, such as demand points to be served by that location, vehicle types and the number of vehicles corresponding to that location etc., grow significantly. Consequently, many more values of variable $y_l$ have to be visited. When *n* increases, only the size of the transportation sub-problem is greater. However, the number of sub-problems is still the same. This larger size of sub-problem does not show a significant effect on the computing time. Moreover, the PTBB1 algorithm uses the lowest computing time to give optimal solutions for all cases.

To express the effect of *u* on the computing time of two parallel approaches, Fig. 3 and Fig. 4 are drawn. According to Fig. 3 and Fig. 4, *u* affects the computing time of both the PTBB1 and PTBB2. The PTBB2 seems to be affected more than the PTBB1 because it has larger gaps between the four lines. In contrast, an increase in *n* at the same *u* does not always increase the computing time. The reason behind these results may be because of the parallel approach. The PTBB2 moves to the next branch, a fixed number of distribution centers, when all workers complete solving the distributed transportation sub-problems. All workers have to wait until the workers that use the longest computing time (bottleneck workers) finish the job. Many small sub-problems distributed in many iterative branches cumulate an enormous waiting time and computing time. Therefore an increase in *u* accelerates computing time of the PTBB2. The PTBB1 alleviates this disadvantage by distributing all transportation sub-problems corresponding to any branch to the same worker and assigning all branches to workers in parallel.

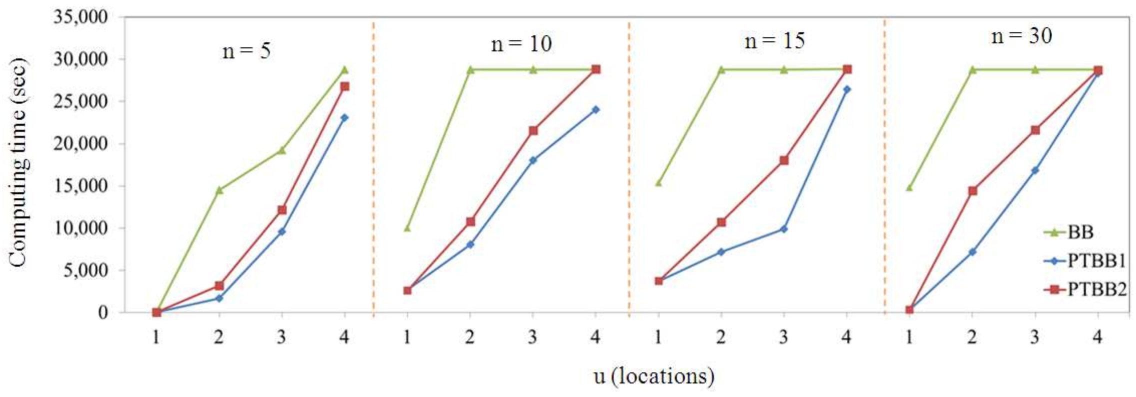

Fig. 2. Graph of the average computing time of BB, PTBB1 and PTBB2 algorithm

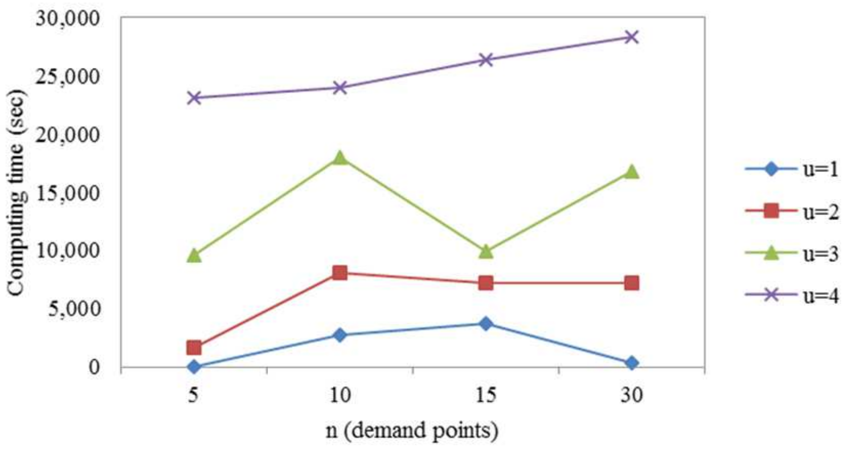

Fig. 3. Graph of average computing time of PTBB1 algorithm

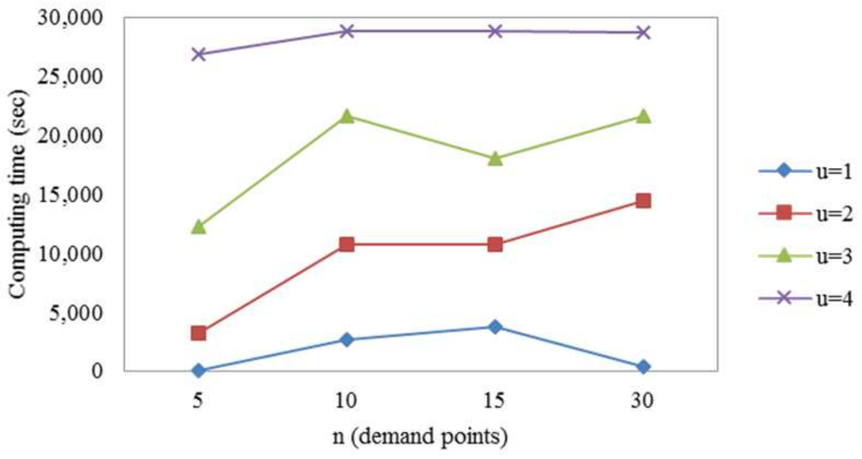

Fig. 4. Graph of average computing time of PTBB2 algorithm

## Conclusion

The sequential BB algorithm can solve the proposed FLTDR model, but it is unable to solve large size problems. Both parallel approaches increase both the efficiency and effectiveness to solve a large size FLTDR. They can solve problem sizes of up to four candidate locations with thirty demand points within 8.01 h. The computing performance depends on both the number of demand points and the number of candidate locations for distribution centers. However, the number of candidate locations for distribution centers has a stronger effect. Moreover, the PTBB2 is affected more than the PTBB1 because of more waiting time between the workers or unbalanced distributed computing tasks. The PTBB1 is recommended for FLTDR because it gives the optimal solution with the least computing time.

To extend the performance of the PTBB algorithms, computer architecture to develop the parallel computing machine should be considered in future research. In addition, heuristic algorithms should be also taken into account as an efficient algorithm for a large size problem ($u \geq 4$). The number of candidate locations reflects the percentage of demand coverage. The greater the percentage of demand coverage required, the greater is the number of distribution centers needed. However, it is limited by resource constraints. Therefore, sensitivity analysis of the vital resource restrictions such as budgetary variables, the number of vehicles, etc. is an interesting area for future research.

## Funding Information

This study was supported by Kasetsart University Research and Development of Thailand under Grant No. 36.51.

## Author's Contributions

All authors equally contributed in this work.

## Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

## References

Abounacer, R., M. Rekik and J. Renaud, 2014. An exact solution approach for multi-objective location-transportation problem for disaster response. Comput. Operations Res., 41: 83-93. DOI: 10.1016/j.cor.2013.08.001

Altay, N. and W.G. Green, 2006. OR/MS research in disaster operations management. Eur. J. Operational Res., 175: 475-493. DOI: 10.1016/j.ejor.2005.05.016

Aslanzadeh, M., E.A. Rostami and L. Kardar, 2009. Logistics Management and SCM in Disasters. In: Supply Chain and Logistics in National, International and Governmental Environment, Farahani, R.Z., N. Asgari and H. Davarzani (Eds.). Physica-Verlag, Heidelberg, ISBN-10: 978-3-7908-2155-0, pp: 221-252.

Bader, D.A., E.H. William, C.A. Phillips, 2005. Parallel Algorithm Design for Branch and Bound. In: Tutorials on Emerging Methodologies and Applications in Operations Research, H J.G, (Eds.). Springer, New York, pp: 5-44. ISBN-10: 978-0-387-22826-6.

Balcik, B. and B.M. Beamon, 2008. Facility location in humanitarian relief. Int. J. Logistics Res. Applicat., 11: 101-121. DOI: 10.1080/13675560701561789

Barreto, L. and M. Bauer, 2010. Parallel Branch and Bound Algorithm-A comparison between serial, OpenMP and MPI implementations. J. Phys. Conf. Ser., DOI: 10.1088/1742-6596/256/1/012018

Boukedjar, A., M.E. Lalami and D. El-Baz, 2012. Parallel Branch and Bound on a CPU-GPU System. Proceedings of the 20th Euromicro International Conference on Parallel, Feb. 15-17, IEEE Xplore Press, Garching, pp: 392-398. DOI: 10.1109/PDP.2012.23

Bourbeau, B., T.G. Crainic and B. Gendron, 2000. Branch and bound parallelization strategies applied to a depot location and container fleet management problem. Parallel Computing, 26: 27-46. DOI: 10.1016/S0167-8191(99)00094-0

Caunhye, A.M., X. Nie and S. Pokharel, 2012. Optimization models in emergency logistics: A literature review. Socio Econ. Planning Sci., 46: 4-13. DOI: 10.1016/j.seps.2011.04.004

Chakroun, I., M. Mezmaz, N. Melab and A. Bendjoudi, 2013. Reducing thread divergence in a GPU-accelerated branch-and-bound algorithm. Concurrency Comput. Practice Experience, 25: 1121-1136. DOI: 10.1002/cpe.2931

Crainic, T.G., B.L. Cun and C. Roucairol, 2006. Parallel Branch-and-Bound Algorithms. In: Parallel Combinatorial Optimization, Talbi, E.G., (Eds.), John Wiley and Sons, USA, pp: 1-26. DOI: 10.1002/9780470053928.ch1

Gendron, B. and T. G. Crainic, 1994. Parallel branch-and-branch algorithms: Survey and synthesis. Operations Res., 42: 1042-1066. DOI: 10.1287/opre.42.6.1042

Leroy, R., M. Mezmaz, N. Melab and D. Tuyttens, 2014. Work stealing strategies for multi-core parallel branch-and-bound algorithm using factorial number system. Proceedings of the Programming Models and Applications on Multicores and Manycores, Feb. 15-19, ACM, USA, pp: 111-119. DOI: 10.1145/2560683.2560694

Minoux, M., 1989. Networks synthesis and optimum network design problems: Models, solution methods and applications. Networks, 19: 313-360. DOI: 10.1002/net.3230190305

Otten, L. and R. Dechter, 2010. Load Balancing for Parallel Branch and Bound. Proceedings of the 10th International Workshop on Preferences and Soft Constraints, Sep. 6, University of Saint Andrews, Scotland, pp: 51-65.

UNISDR, 2012. The Economic and Human Impact of Disasters in the last 12 years. United Nations Office for Disaster Risk Reduction.