An Efficient Guided Local Search Approach for Service Network Design Problem with Asset Balancing

Ruibin Bai[1], Graham Kendall[2] , Jiawei Li[2]

1. Division of Computer Science, University of Nottingham Ningbo, China, 315100
E-mail: ruibin.bai@nottingham.edu.cn

2. School of Computer Science, University of Nottingham, Nottingham, UK, NG8 1BB
E-mail: gxk/jwl@cs.nott.ac.uk

Abstract: Service network design is a core problem for logistic transportation planning. It involves determination of the most cost-effective transportation network, package flow distribution as well as balanced vehicle schedules. In this paper, we propose an efficient guided local search approach metaheuristic for this problem, which is able to produce competitive results with much less computational time than those proposed in the literature.

Key Words: Logistics, Freight Transportation, Guided Local Search, Linear Programming

I. INTRODUCTION

Service network design is widely used as a generic model to formulate network design problems faced in the fields of logistics, telecommunication, transportation and production systems. The problem is mainly concerned with searching a network configuration that minimizes the total costs incurred for shipping customers' commodities through networks of limited capacity. Depending on the application, the costs may include the fixed costs that must be invested for opening (or building) an arc, it can also involve variable costs that are proportional to the amount of commodities to be routed. Two of the most common constraints in the service network design problem are the network capacity constraint and flow conservation constraint (often referred as the fixed-charge capacitated multi-commodity network design problem, or CMND in abbreviation) [7]. The CMND model is related to the classical network flow problem [1] but is much more difficult to solve. It has been proved that CMND problems are NP-hard [7]. Reports also suggests that the state-of-the-art exact solvers struggle to handle CMND problem cases of any realistic size [9,4,7]. There are some successful applications where some large real-world oriented instances have been solved successfully by exploiting special structures of the problem and using more sophisticated modeling techniques [2,3]. However, it may be difficult to generalize the approach to service network design problems without similar structures or characteristics.

The CMND model can be augmented to adapt to different problem scenarios and requirements. One of modifications in freight transportation is the inclusion of an asset-balance constraint [4,10], which is essential for freight transportation companies to ensure the continuity of the services over time. That is, at the end of each service time window, the amount of transportation assets (e.g. vehicles, containers, etc.) should be kept at the same level as the previous level.

Scheduling the transportation assets are generally based on a time-space network where a planning horizon (e.g. a week) is discretized into several time slots (e.g. a day). In the time-space network, each physical node (or terminal) has a copy in every time slot. For a network with *n* physical nodes and *m* periods, the number of nodes in the time-space network will increase significantly to *mn*. Figure I describes a simple time-space network with 3 terminals and 7 time slots. Two schedules are displayed in this figure. The schedule represented by solid lines indicates that a vehicle leaves from node 3 on day 2 and arrives at node 2 and node 1 on day 3 and day 4 respectively. It then returns to node 3 on day 5 and stays at node 3 until day 2 of the next planning cycle. Similarly the schedule represented by the dashed lines specifies a vehicle movement path from node 1 to node 2 on day 1 and then returning to node 1 on day 6. It then moves to node 2 on day 7 and stays at node 2. For this simple example, we say that node 3 is asset-balanced since the incoming vehicles and outgoing vehicles are the same throughout the planning horizon. However, for node 1 and node 2 this is not the case since neither of them is balanced on day 1.

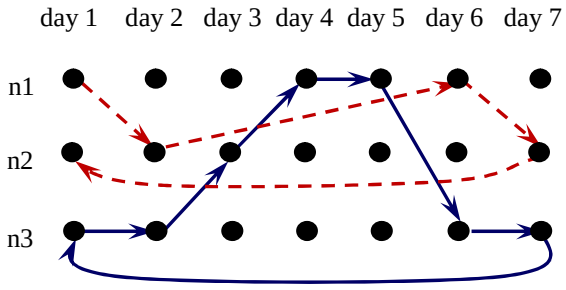

Figure I A time-space network with three nodes.

Considerable progresses have been made in terms of investigating and developing effective meta-heuristic or hybrid algorithms for CMCD problems, in particular, tabu search and path-relinking have been thoroughly investigated and improved, with very good computational results having been obtained [5,7,8,10]. These approaches are generally competitive against the state-of-the-art MIP solvers for small and medium sized problems and more importantly demonstrates superior capability when dealing with large problem instances. Unfortunately, although these heuristic based approaches are much faster than the exact MIP solver for large problem instances, from a practical point of view the amount of computational time they take is still quite large. For example, in [10] a single run of the tabu search approach required 3600 seconds on a machine with Pentium IV 2.26GHz CPU. This paper intends to improve the

efficiency of the current metaheuristics for CMND problems by investigating other heuristic based approaches. It is hoped that the proposed algorithm is able to produce similar quality results but with much less computational effort.

The paper is organized as follows: in the next section, we present a formal formulation of the asset-balanced CMND problem and its associated capacitated multicommodity min-cost flow problem (CMMCF). Section 3 describes the guided local search algorithm proposed in this paper and its performance is discussed and analysed in section 4. Section 5 concludes the paper.

II. PROBLEM AND FORMULATIONS

Pedersen et al. [10] introduced a node-arc based "design-balanced capacitated multicommodity network design" (DBCMND) formulation. This formulation represents a generic model for service network design. For completeness, we also present it here. Let $G=(N, A)$ denote a service network with a set of nodes N and a set of arcs A. Denote $(i, j) \in A$ be the arc from node i to node j. Let K be the set of commodities to be transported via the network. For each commodity $k \in K$, let $o(k)$ and $s(k)$ stand for its origin and destination respectively. Other notations used in the model can be found from Table I.

Table I. List of notations used in the formulation

y_{ij}	Boolean design variables. y_{ij} equals 1 if arc (i, j) is used in the final design and 0 otherwise.
x_{ij}^k	is flow of commodity k on arc (i, j).
$N^+(i)$	The set of outward neighbours of node i, i.e. $N^+(i)=\{j \mid$ and $(i, j) \in A\}$.
$N^-(i)$	The set of inward neighbours of node i, i.e. $N^-(i)=\{j \mid j \in N$ and $(j, i) \in A\}$.
u_{ij}	Capacity of arc (i, j).
f_{ij}	Fixed cost of arc (i, j).
c_{ij}^k	Variable cost of moving one unit of commodity k along arc (i, j).

For simplicity, denote Y and X the design variable vector and flow distribution variable vector respectively. The DBCMND model can then be formulated as:

$$\min \quad z(X,Y) = \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{k\in K} \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k \qquad (1)$$

subject to

$$\sum_{k\in K} x_{ij}^k \le u_{ij} y_{ij} \quad \forall (i,j) \in A \qquad (2)$$

$$\sum_{j\in N^+(i)} x_{ij}^k - \sum_{j\in N^-(i)} x_{ji}^k = b_i^k \quad \forall i \in N, \forall k \in K \qquad (3)$$

$$\sum_{j\in N^-(i)} y_{ji} - \sum_{j\in N^+(i)} y_{ij} = 0 \quad \forall i \in N \qquad (4)$$

$$x_{ij}^k \ge 0 \quad \forall k \in K, \forall (i,j) \in A \qquad (5)$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \qquad (6)$$

where b_i^k is defined as

$$b_i^k = \begin{cases} d^k & \text{if } i = o(k) \\ -d^k & \text{if } i = s(k) \\ 0 & \text{otherwise.} \end{cases} \qquad (7)$$

The objective is to minimize the sum of the fixed cost of the selected arcs and the total variable shipping costs. Constraint (2) ensures that the total amount of flow on an arc does not exceed the limit, determined by the value of design variables. Constraint (3) is the *flow conservation constraint*. Constraint (4) is the *asset-balance constraint* which ensures transportation assets (e.g. vehicles) are balanced at the end of each planning period. Constraints (5) and (6) ensure non-negativity of commodity flows and binary design variables.

There is a natural decomposition of the problem into two related sub-problems: 1). determination of optimal design variables $\bar{Y} = \{\overline{y_{ij}} \mid (i,j) \in A\}$ firstly, and then, 2). for a given feasible design variable vector $\bar{Y}$, searching the optimal flow distribution. The second sub-problem can be solved as follows. Set the flow variables $x_{ij}^k = 0$ for all arcs $(i,j) \notin \bar{A}$ where $\bar{A}$ is the set of open arcs in the design vector $\bar{Y}$. Flow distribution variables for all open arcs (i.e. $(i,j) \in \bar{A}$) can be obtained by solving the following capacitated multicommodity min-cost flow problem (CMMCF).

$$\min \quad \bar{z}(X) = \sum_{k\in K} \sum_{(i,j)\in \bar{A}} c_{ij}^k x_{ij}^k \qquad (8)$$

$$\sum_{k\in K} x_{ij}^k \le u_{ij} \quad \forall (i,j) \in \bar{A} \qquad (9)$$

$$\sum_{j\in N^+(i)} x_{ij}^k - \sum_{j\in N^-(i)} x_{ji}^k = b_i^k, \quad \forall i \in N, \forall k \in K \qquad (10)$$

$$x_{ij}^k \ge 0 \quad \forall k \in K, \forall (i,j) \in \bar{A} \qquad (11)$$

III. OUR PROPOSED APPROACH

In this paper, we propose to use a guided local search algorithm in combination with a linear programming solver (we use lp_solve in this paper). It is hoped that the proposed algorithm is more efficient than those that have been investigated so far in the literature [6-8,10]. Most metaheuristic approaches adopted the same decomposition as we mentioned in section II. That is, metaheuristic approaches are used to search for a good network design vector $\bar{Y}$, and for a given network design vector $\bar{Y}$, an LP solver is then used to obtain the optimal flow distribution by solving the corresponding CMMCF problem. Therefore, these types of approaches involve solving the CMMCF problem many times which, unfortunately, is computationally expensive. To speed up the search, one has to keep the number of LP solver calls to a minimum and design local search approaches to search the neighborhood more efficiently. For example, the tabu search method implemented in [10] evaluates a subset of the neighborhood only based on some heuristics. Although the algorithm outperforms a state-of-the-art MIP solver, the amount of computational time required for a single run of the algorithm is still very large (an hour) and impractical for real-world day-to-day applications.

One of the main advantages of the guided local search method is its ability to exploit problem specific features to guide the search towards promising regions. More specifically, in a guided local search method, a set of features are carefully selected with presence of "bad" features in a solution is punished according to a dynamically tuned function so that the search is guided towards regions containing "attractive" features and avoid less promising regions. It is hoped that the guided local search either solves the problem with much less computational effort than the previous tabu search method, or, given the same amount of computational time, the proposed algorithm obtains better results.

input: an initial solution s_0, an original objective function $g(s)$, a set of features R, costs associated with each feature $h_r (r \in R)$ and a scaling parameter λ.
begin
  foreach $r \in R$, set $p_r := 0$;
  $E(s) = g(s) + \lambda \times \sum_r p_r I_r(s)$;
  while stopping criterion is false
    $s \leftarrow LocalSearch(s, E(s))$
    foreach $r \in R$ **do**
      $util_r(s) = I_r(s) \times \frac{h_r}{1+p_r}$
      Find r with maximum $util_r$, set p_r++;
    end foreach
  end while
  return $s^* \leftarrow$ best solution found according to function g;
end

Figure II Pseudo-code for a basic guided local search procedure.

A. Guided Local Search

Guided local search (GLS) is a metaheuristic designed for constraint satisfaction and combinatorial optimization problems [11]. The core of the guided local search method is the identification of a set of features and determination of a transformed evaluation function. The transformed evaluation function for a given solution *s* appears in the form of:

$$E(s) = g(s) + \lambda \times \sum_r (p_r \times I_r(s)) \qquad (12)$$

where $g(s)$ is the original objective function. p_r is the penalty for a given feature *r* existing in the current solution *s*. $I_r(s)$ is an indicator variable: $I_r(s)=1$ means the candidate solution contains feature *r* and $I_r(s)=0$ otherwise. The penalty value p_r for each feature can be dynamically changed. The selection of features to be penalized is based on a utility value, $util_r(s)$, defined as $util_r(s) = I_r(s) \times h_r / (1+p_r)$, where h_r are the cost of the feature *r*. λ is a control parameter which is often estimated by $\lambda = \alpha \cdot g(s^*) / \sum_r I_r(s^*)$ where s^* is the current local optimum and α is a parameter that is less problem-dependent than λ. The basic GLS approach can be illustrated by Figure II.

In this application, we choose all arcs be the GLS features and their fixed costs be the features costs, i.e. $h_{ij}=f_{ij}$. Alternatively, one can introduce a feature cost function that reflects both the fixed cost, variable cost and the popularity of the arc. The drawback is that it introduces more parameters into the algorithm.

B. Constraint Handling

Network capacity constraint (2) and flow conservation constraint (3) are handled directly in the local search. That is, any moves that violate them will be discarded. However, the asset-balance constraint (4) is relaxed and violations of this constraint are allowed but are penalized according to the following function:

$$g(X,Y) = z(X,Y) + \tau \times \bar{f} \times \sum_{i \in N} |\psi_i|^{\gamma} \qquad (13)$$

where $\bar{f}$ is the average of the fixed costs of the arcs in the network. ψ_i, *node asset-unbalance*, stands for the difference (or unbalance) between outgoing open arcs and incoming open arcs at node *i*, i.e. $\psi_i = \sum_{j \in N^+(i)} y_{ij} - \sum_{j \in N^-(i)} y_{ji}$. τ is a scaling parameter that controls the importance of the penalty term. Note that the node asset-unbalance was magnified by a power γ(> 1) in order to apply higher penalties to highly unbalanced nodes. In this paper we set γ =2 for ease of computation. Note that this penalty function is slightly easier to compute than the one used in [10].

C. Neighborhood Definition

The neighborhood definition used in the guided local search is similar to the one used in [10]. From an incumbent solution, a neighbor solution is generated by either closing an open arc or opening a closed arc. Closing an arc could result in a better solution since the fixed cost of this arc will be removed from the objective function. Meanwhile, opening an arc can also lead to an improvement in the evaluation function (13) since it could reduce node unbalance.

1) Closing Arcs: To close an arc (*i*, *j*) that has a positive flow, one needs to redirect the flow to the remaining open arcs. In theory, the optimal flow re-distribution can be obtained by solving the model (8)-(11). Unfortunately, this would be computationally too expensive. To alleviate computational burden, similar to the most previous approaches, a heuristic method based on a *residual network* [7] and the Dijkstra shortest path algorithm is used. More specifically, let $resCap_{lt}$ denote the residual capacity of the arc (*l*, *t*) $\in A$ in the current solution ($resCap_{lt}$ defaults to u_{lt} if the arc is closed). For each commodity *k* that has a positive flow on the arc (*i*, *j*), a Γ^k residual network $G^{\Gamma^k} = (N, A^{\Gamma^k})$, where $\Gamma^k = d_k$, is constructed with the arcs in this residual network defined as:

$$A^{\Gamma^k} = \{(l,t) \in A \,|\, (l,t) \neq (i,j) \wedge resCap_{lt} \geq \Gamma^k\}.$$

We compute the "cheapest" path on this residual network and redirect entire flow of the commodity *k* to this single path. If such a path cannot be found, the move is considered infeasible and the search goes back to the incumbent solution. The cost associated with each arc in this residual network is defined as:

$$c_{lt}^{\Gamma} = \begin{cases} f_{lt} + c_{lt} \cdot d^k & \text{if } (l,t) \in A^{\Gamma^k} \text{ is closed,} \\ c_{lt} \cdot d^k & \text{if } (l,t) \in A^{\Gamma^k} \text{ is open.} \end{cases}$$

The above flow redistribution is done for each commodity that has a positive flow on the arc (*i, j*). If the procedure fails to redistribute the flow for any of them, this arc closing procedure is terminated and the move is considered infeasible.

2) Opening Arcs: Although opening an arc will increase the objective value due to the fixed cost of the arc, it could potentially reduce the node unbalance penalty and hence lead to more feasible solutions. When a closed arc is opened, the optimal flow distribution is probably changed as well. One would have to solve the CMMCF model in order to obtain this optimal flow distribution. However, as stated earlier, it is computational prohibitive to solve this model for every possibility. Therefore, similar to [10], the flow distribution is kept the same and the fixed cost of this arc is incurred.

3) *Opening a path*: For two nodes *i, j* with opposite asset unbalance (i.e. $\psi_i \cdot \psi_j < 0$), opening or closing a path between
i and *j* will reduce the asset unbalance penalty for both nodes (note that the direction of the path will depend on the actual signs of ψ_i and ψ_j). If, however, there is no direct arc linking between the nodes, a neighborhood move of closing/opening a single arc will not reduce the node asset unbalance. It is benedictory to open a path between the nodes. In addition, sometimes when the local search stagnates at a local optimum, it is useful to randomize the incumbent solution and restart the search to escape from this local optimum. Therefore, we also introduced another neighborhood operator, which selects a random commodity and opens one of its top 10 shortest paths with equal probability. The corresponding CMMCF model is then solved to obtain the optimal flow distribution. Since this operator is only called occasionally, solving the CMMCF problem in this procedure will have much less impact on the speed of the algorithm. The top 10 shortest paths for a given commodity are computed independently without consideration of the other commodities. Therefore, they are computed only once at the beginning of the search.

D. Recovering Feasibility

Although relaxing the asset-balance constraint will provide freedom and flexibility in designing more effective neighborhoods. For most instances, the algorithm converges to a feasible solution (i.e. all nodes are asset-balanced). There are, however, some instances that the algorithm is struggling to find a high quality feasible solution. Therefore, a specialized heuristic is required to repair the solution and recover the feasibility. In this paper, we used a similar heuristic introduced by [10]. The main idea of this heuristic is to repeatedly reduce the asset-unbalance of the most unbalanced node (i.e. $|\psi|$ is maximum). This is achieved by closing (or opening) a path between this node and another unbalanced node with opposite ψ sign. Since there are potentially many paths between these two nodes, the heuristic only evaluates four shortest paths, computed by solving the shortest path problems in the four different modified networks. Again since opening a path involves opening/closing several arcs, the CMMCF model is solved after each neighborhood move. More details can be found in [10].

begin
  Generate an initial solution s_0 by rounding design variables of the LP solution of model (1)-(6);
  Set $s' := s_0$; $s^* = s_0$;
  while TimeAvailable **do**
    /* Guided Local Search Phase */
    while TimeAvailable && $nonImpCount < K$ **do**
      Set $s' := s$;
      foreach arc $(i,j) \in \mathcal{A}$ **do** $\{p_{ij} := 0;\ h_{ij} := f_{ij};\}$
      $s \leftarrow BestNeighbour(s, E(s))$;
      $s \leftarrow CMMCF(s)$;
      if $g(s) < g(s')$ set $s' := s$;
      Update the best feasible solution s^* with respect to $z(X,Y)$;
      /* Update GLS Parameters */
      foreach arc in $\mathcal{A}$ **find** the arc (i,j) that maximises $util_{ij} = I_{ij}(s') \cdot h_{ij}/(1+p_{ij})$;
      Set $p_{ij} := p_{ij} + 1$;
    end while
    /* Feasibility Recovery Phase */
    $s \leftarrow FeasibilityRecovery(s')$
    Update the best feasible solution s^*;
    $s := Perturbation(s')$;
  end while
end
return s^*.

Figure III Pseudo-code A multi-start guided local search algorithm for AB-CMND.

The proposed algorithm is shown in Figure III. The initial solution is built by solving the relaxed LP model (i.e. constraint (6) is relaxed) and then rounding the design variables to binary. The algorithm mainly consists of two phases: the guided local search phase and the feasibility recovery phase. These two phases are executed repeatedly until the computational time is exhausted. The guided local search phase stops when either no computational resource is left or the number of consecutive non-improving iterations reaches a given value *L*. In the guided local search, the neighborhood is defined by closing/opening arcs as described in section III-C. Procedure BestNeighbour(*s*, *E*(*s*)) returns the best neighbor of the current solution *s* according to the augmented objective function *E*(*s*). Since the flow distribution is estimated in neighborhood exploration, the solution returned by BestNeighbour(*s*, *E*(*s*)) is re-optimised by solving the corresponding CMMCF model. During the guided local search phase, the best solution with regard to *g*(*s*) and the best feasible solution with regarding to *z*(*X*, *Y*) are recorded.

Since the guided local search operates on a transformed evaluation function *E*(*s*), a "good" solution according to *E*(s) and *g*(*s*) does not necessarily represent a "good" feasible solution in terms of *z*(*X*, *Y*). Therefore, procedure FeasibilityRecovery(*s*) is applied after the guided local search phase to recovery the solution feasibility (i.e. asset-balance) using the heuristic described in section III-D. If this phase finds a solution that is better than the best

feasible solution in terms of the original objective $z(X, Y)$), the best feasible solution is updated.

After FeasibilityRecovery(s) finishes, the solution is perturbed using the path-opening heuristic (see section III-C3) to get ready for the next round guided local search. The algorithm stops when the computational time is exhausted.

IV. Experiment Set Up and Results Analysis

The proposed multi-start GLS algorithm is implemented in Microsoft Visual C# 2008. The algorithm is run on a PC with Intel Core 2 CPU 1.8GHz and 1GB RAM. However, since the program is sequential, only one processor is utilized. After some preliminary tests, we fixed the parameters of the algorithm as follows: L=30, τ =0.5, α =0.2 and the maximum time allowed for a single run of the algorithm is set to 2400 seconds, a 1/3 reduction in computational time compared with the tabu search approach in [10].

Table II Font Settings Computational results for the C problem instances by different algorithms.

Instance	Xpress MIP	TS (1 run)	GLS (5 runs) best	GLS (5 runs) average
C20,230,200,V,L	101112	102919	101267	**102324**
C20,230,200,F,L	153534	150764	143017	**146584**
C20,230,200,V,T	105840	**103371**	103428	104090
C20,230,200,F,T	154026	149942	143446	**144941**
C20,300,200,V,L	81184	82533	80183	**80694**
C20,300,200,F,L	131876	128757	126097	**127334**
C20,300,200,V,T	78675	**78571**	77839	78910
C20,300,200,F,T	127412	**116338**	116712	120080
C30,520,100,V,L	55138	55981	55437	**55548**
C30,520,100,F,L	n/a	104533	100999	**102462**
C30,520,100,V,T	53125	54493	53644	**53809**
C30,520,100,F,T	106761	**105167**	106096	106972
C30,520,400,V,L	n/a	**119735**	119344	121441
C30,520,400,F,L	n/a	**162360**	163182	164146
C30,520,400,V,T	n/a	**120421**	122877	374020
C30,520,400,F,T	n/a	**161978**	166488	167146
C30,700,100,V,L	48849	**49429**	49465	49532
C30,700,100,F,L	65516	63889	62936	**63685**
C30,700,100,V,T	47052	48202	47518	**47646**
C30,700,100,F,T	57447	**58204**	58559	59063
C30,700,400,V,L	n/a	**103932**	104534	104826
C30,700,400,F,L	n/a	157043	152580	**152842**
C30,700,400,V,T	n/a	103085	**103581**	413682
C30,700,400,F,T	n/a	141917	**142575**	143876

Benchmark datasets from the literature [7], [10] are used as a testbed for the proposed algorithm. The instances we used are drawn from the two sets (C, R) with each set containing problem instances of different sizes (nodes, arcs, commodities) and distributions of fixed cost, variable cost and capacity. The first three numbers in the instance name represent the number of nodes, the number of arcs and number of commodities respectively. A letter 'F' means that the fixed costs are important relative to the variable costs while 'V' stands for dominant variable costs. Letter 'L' stands for loose capacity constraints while 'T' means capacities are tight. For each instance, 5 independent runs were carried out. In Table II, we provides both the best results and the average results by our proposed algorithm, in comparison with those obtained by Xpress-MP MIP solver and the tabu search method by Pedersen et al. [10].

Table II shows the results for C set instances (R set instances are not included here due to space reasons). We can see that the proposed algorithm produces promising results, in comparison with the Xpress MIP solver and the tabu search approach recently proposed. Since results by TS are based on a single run, for fairness, we compare against the average results of our proposed guided local search. For most of cases, the guided local search performs competitively when compared with TS. It even outperforms TS for 11 out of 24 instances. This is a very good achievement considering that GLS used much less computational time on a slower processor (TS was run on a PC with 2.4GHz processor while GLS was run on a PC with 1.8GHz processor). For two instances (C55 and C63) the average results by GLS were poor. Nevertheless, the best results by GLS are still competitive. It seems that the proposed algorithm struggles to perform consistently for some large sized instances. Finally, Xpress MIP obtained the best results for some instances but failed to return a feasible solution for 9 instances (marked as "n/a").

Similar results have been obtained for the R set instances (consisting of 53 instances) except for 4 instances where the proposed algorithm fails to find a feasible solution. A close examination of the results shows that lp_solve fails to solve the CMMCF problem in reasonable time. For space reasons, we could not include results in this paper but they are available on request.

V. Conclusions and Future Work

In this paper, we proposed an efficient approach that can be used to optimise the service network design problem in the logistic freight transportation. The proposed algorithm incorporates a guided local search metaheurisic within a multi-start framework. Computational results have demonstrated the fast convergence ability of GLS metaheuristic and its ability to produce promising results with much less computational effort, when compared with a recently proposed tabu search approach. Future research will focus on incorporating new mechanisms to improve the robustness of the algorithm.

References

[1] Andrew P. Armacost, Cynthia Barnhart, and Keith A. Ware, Composite variable formulations for express shipment service network design, Transportation Science, Vol.36, No.1, pp. 1–20, 2002.

[2] Andrew P. Armacost, Cynthia Barnhart, Keith A. Ware, and Alysia M. Wilson, Ups optimizes its air network, Interfaces, Vol. 34, No. 1, pp.15–25, 2004.

[3] Cynthia Barnhart, Niranjan Krishnan, Daeki Kim, and Keith Ware, Network design for express shipment delivery, Computational Optimization and Application, Vol. 21, pp.239–262, 2002.

[4] Teodor Gabriel Crainic, Michael Gendreau, and Judith M Farvolden, A simplex-based tabu search method for capacitated network design, INFORMS Journal on Computing, Vol.12, No. 3, pp. 223–236, 2000.

[5] Teodor Gabriel Crainic, Ye Li, and Michel Toulouse, A first multilevel cooperative algorithm for capacitated multicommodity

network design, Computers & Operations Research, Vol. 33, No. 9, pp. 2602–2622, 2006. Part Special Issue: Anniversary Focused Issue of Computers & Operations Research on Tabu Search.

[6] Ilfat Ghamlouche, Teodor G. Crainic, and Michel Gendreau, Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design, Operations Research, Vol. 51, No. 4, pp. 655–667, 2003.

[7] Ilfat Ghamlouche, Teodor G. Crainic, and Michel Gendreau, Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design, Annals of Operations Research, Vol. 131, pp. 109–133, 2004.

[8] Deaki Kim, Large scale transportation service network design: models, algorithms and applications, PhD thesis, Massachusetts Institute of Technology, Boston, USA, 1997.

[9] Michael B. Pedersen, Teodor G. Crainic, and Oli B.G. Madsen, Models and tabu search metaheuristics for service network design with asset-balance requirements. Transportation Science, Vol. 43, No. 2, pp. 158–177, 2008.

[10] Christos Voudouris and Edward P.K. Tsang, Guided local search. In

[11] F. Glover and G. Kochenberger, Handbook of Metaheuristics, 185–218. Kluwer, 2003.