Genetic Algorithm for the Transportation Problem with Discontinuous Piecewise Linear Cost Function

***Su Sheng*[†], *Zhan Dechen*[†], *and Xu Xiaofei*[†],**

[†] *School of computer science and technology, Harbin Institute of Technology, Harbin, China*

Summary
We investigate a transportation problem with discontinuous piecewise linear cost function in this paper. The basic feasible solution for the transportation problem, in which flow bounds on edges are uncertain, is obtained by disaggregating piecewise linear cost function. A genetic algorithm is proposed to solve the transportation problem based on the network representation of the basic feasible solution. A basic feasible solution is represented by the spanning tree that consists of all basic edges and nonbasic edges with positive flow amount. Sorted edge set is used to represent the spanning tree. Edges of a spanning tree are sorted by root first search pattern in the representation. Single point crossover operator and random pivot mutation operator are developed for effective and efficient evolution. Computational tests demonstrate our genetic algorithm not only can obtain better solution quality, but also is more efficient than the previous proposed genetic algorithm based on the matrix encoding.

Key words:
Transportation problem; discontinuous piecewise linear cost function; genetic algorithm; spanning tree

1. Introduction

The transportation problem with discontinuous piecewise linear cost function (the TPDPLC problem) arise in many practical applications areas. Some transportation problems with discount consideration have this kind of cost function, and some production planning and scheduling problems where production amount need to be divided into several small portions or setup time need be considered can also be modeled as this kind of transportation problem. Other applications can be found in telecommunications and logistics domain, and so on.

Though it can be found in many applications, research on transportation problems with discontinuous piecewise linear cost function is not so rich. Transportation problems with discontinuous piecewise linear cost function can easily be formulated and solved as a mixed integer programming problem. Theoretically, any general mixed integer programming solution method can be used to solve this kind of problem, for example, branch and bound method, branch and cut method. However, these methods are generally inefficient and computationally expensive for the problems with large size.

Metaheuristic methods, can obtain global optimal solution or approximately global optimal solution at the cost of less computational time, are preferable in practical industry applications. To the best of our knowledge, the matrix encoding genetic algorithm for generic nonlinear transportation problems developed by Michalewicz, et al. [1] is an only metaheuristic method relevant directly to our problem. They developed a genetic algorithm for the transportation problem with general nonlinear cost functions and demonstrated its superiority relative to off-the-shelf nonlinear programming solvers, such as GAMS, when cost functions are not smooth. However, solution quality and efficiency of the matrix encoding genetic algorithm are not so ideal, as an evolutionary algorithm. To improve solution quality and decrease computational time, we propose a genetic algorithm for transportation problems with discontinuous piecewise linear cost function based on the cost function disaggregation.

Although the research on transportation problems with discontinuous piecewise linear cost function are lack, there are extensive relevant literatures on transportation problems. The first kind of relevant work is the fixed charge transportation problem. The fixed charge transportation problem is simplified version of the TPDPLC problem whose cost function has only one segment. The fixed charge transportation problem has been studied by many researchers from about four decades years ago. A number of exact solution algorithms are developed to solve the FCT problem, which include cutting plane approaches [2], vertex ranking methods [3, 4], and branch and bound approaches [5, 6]. Some heuristic methods are also proposed [7, 8]. Recently, some metaheuristic methods are developed for the FCP problem. Sun et al. [9, 10] developed a tabu heuristic search procedure. Gottlieb and Paulmann proposed a matrix representation genetic algorithm [11]. Li et al. (1998) proposed a genetic algorithm where prüfer numbers are use to encode spanning tree [12].

The second kind of relevant work is network flow problems with piecewise linear cost functions. Network flow problem, where transportation network has several stages, is extension of transportation problem. Although

Manuscript received July 5, 2006.
Manuscript revised July 25, 2006.

the literatures on network flow is rich, research concerning problems with piecewise linear cost functions is limited. Piecewise linear cost functions can be classified two categories. One category is piecewise linear convex or concave function. Balakrishnan and Graves develop a Lagrangian heuristic algorithm for the uncapacitated network flow problem with piecewise linear concave cost [13]. Aghezzaf and Wolsey model piecewise linear concave costs in a tree partitioning problem [14]. Cominetti and Ortega study minimum concave cost network flows [15]. They propose a branch and bound algorithm and improve the approximations and obtain better lower bound using sensitivity analysis. Chan et al. study the minimum concave cost multi-commodity network flow problem, and characterize structural properties of the optimal solution of the linear programming relaxation and propose a heuristic solution approach that uses these properties to transform the fractional solution into an integer one [16].

Another category is nonconvex and nonconcave cost function. Breakpoints occur generally between any two adjacent segments in the function, as studied in the paper. Croxton et al. examine a multi-commodity network flow problem with this kind of function [17]. They exploit the disaggregation techniques to solve a logistics problem called the merge-in-transit problem. Kim and Pardalos propose a dynamic domain contraction algorithm for the nonconvex piecewise linear network flow problem, which iteratively solves a series of linear network flow problems with dynamically adapted costs and restricted domains [18]. Gabrela et al. develop an exact algorithm for multicommodity network optimization with general step cost functions [19]. The proposed procedure may be viewed as a specialization of the well-known Benders partitioning procedure, leading to iteratively solving an integer 0-1 linear programming relaxed subproblem which is progressively augmented through constraint generation.

The third kind of relevant work is the facility location and network design problems. Holmberg develops algorithms for the facility location problem with staircase cost using Bender's decomposition and Lagrangian heuristic methods [20, 21]. However, most of the research in the areas of facility location and network design has focus on the fixed charge case. Blakrishnan et al. provide an annotated bibliography concerning network design problems up to 1997 [22]. Crainic et al. proposed a bundle-based relaxation method for capacity fixed charge network design problems [23]. Magnanti et al. model and solve a two facility capacitated network loading problem [24].

2. Problem statement

Given a transportation network $G = (V, E)$ where V is the node set consisting of all source nodes and destination nodes, E is the arc set. Let m denote the number of source nodes, n the number of destination nodes, C_i the capacity of source node i, D_j the demand of destination j, (i, j) the arc from source node i to destination j, $x_{i,j}$ the flow amount on arc (i, j) (the decision variable). As shown in Fig. 1, we define the cost $g_{i,j}(x_{i,j})$ on arc (i, j) as a piecewise linear increasing function with a finite set of discontinuities. In addition, we assume that $g_{i,j}(0) = 0$.

Discontinuous piecewise linear function can be characterized by its segments. To any arc (i, j), each segment s has a nonnegative variable cost $v_{i,j,s}$ (the slope), a nonnegative fixed cost $f_{i,j,s}$ (the intercept), upper bound $ub_{i,j,s}$ of the flow and lower bound $lb_{i,j,s}$ of the flow. Each arc (i, j) has a finite number of segments because the total flow on arc (i, j) can be bounded the minimum value between the capacity of node i and the demand of destination j. we denote segments on arc (i, j) by the set $S_{i,j} = \{1, 2, \ldots, NB_{i,j}\}$. Let $lb_{i,j,1} = 0$. It can be inferred that the problem is more difficult to solve if there are more segments on arc (i, j). With these notations, we can express the transportation problem with discontinuous piecewise linear cost as the following formulation:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} g_{i,j}\left(x_{i,j}\right)$$

$$s.t. \quad \sum_{j=1}^{m} x_{i,j} = C_i, \qquad i = 1,2,\cdots,m \tag{1}$$

$$\sum_{i=1}^{n} x_{i,j} = D_j, \qquad j = 1,2,\cdots,n \tag{2}$$

$$x_{i,j} \geq 0, \qquad i = 1,2,\cdots,n;\ j = 1,2,\cdots,m \tag{3}$$

where

$$g_{i,j}\left(x_{i,j}\right) = \begin{cases} 0 & x_{i,j} = 0 \\ f_{i,j,s} + v_{i,j,s} x_{i,j} & lb_{i,j,s} < x_{i,j} \leq ub_{i,j,s} \end{cases} \tag{4}$$

In the above model, $s \in S_{i,j}$, equation (1) and (2) is the capacity balance and the demand balance, respectively. The sum of flows on all arcs is related to source i (i = 1, 2, …, m)should be equal to the capacity of source i, C_i, on the right hand side of equation (1). The sum of flows on all arcs is related to destination j (j = 1, 2, …, n)should be equal to the demand of destination j, D_j, on the right hand side of equation (2). Inequality (3) restricts the flow amount on any arc must be larger than or equal to zero. Equation (4) defines the relationship between the flow amount on arc and cost. The objective function is the minimizing of the total transportation cost on all arcs.

The TPDPLC problem is simplified to the fixed charge transportation problem if the number of segment is decreased to one ($NB_{i,j} = 1, \forall\, (i, j)$). The equation (4) will be substituted by the following equation (5) and constraints (1)~(3) hold for the fixed charge transportation problem.

$$g_{ij}\left(x_{ij}\right)=\begin{cases}0 & x_{i,j}=0\\ f_{i,j}+v_{i,j}x_{i,j} & x_{i,j}>0\end{cases} \tag{5}$$

where $f_{i,j}$ is the fixed cost on arc (i,j). Since the fixed charge transportation problem version of the TPDPLC problem is known to be *NP*-hard, the TPDPLC problem is *NP*-hard as well. The paper [28] proved that the optimal solution for the fixed charge transportation problem occurs at one of extreme points defined by constraint in Equation (1)~(3). An extreme point is a basic feasible solution of the fixed charge transportation problem, which has m+n-1 basic variables and they must be a spanning tree of the transportation network. However, since the cost function is nonconvex and nonconcave for the TPDPLC problem, the optimal solution of the TPDPLC problem need not be an extreme point of the convex set defined by constraints. It can occur within the convex set. Let X be the set of all feasible transportation assignment S.

Definition 1. Let set $B_k = \{S \mid S \in X,\ lb_{i,j,s} \le x_{i,j} \le ub_{i,j,s}\}$, where k is an mn vector with $((i\text{-}1)n+j)$th component $k_{i,j}$, and $k_{i,j}$ can be any value of set $S_{i,j} = \{1, \cdots, NB_{i,j}\}$, and subindex $s \in S_{i,j}$. B_k is called a basic set.

The TPDPLC problem defined on a certain B_k can be seen as a fixed charge transportation problem with lower bound and upper bound constraints of the flow on each arc. The basic feasible solution of the TPDPLC problem on B_k is denoted by $BF_k \in (F_k, L_k, U_k)$ according to the network simplex method, where F_k is the set of m+n-1 basic arcs, L_k is the set of arcs whose flow are equal to lower bounds, and U_k is the set of arcs whose flow are equal to upper bounds.

Definition 2. Let $E[B_k]$ be extreme points of basic set B_k, we call the set $D = \cup_k E[B_k]$ the dominant set.

Then, the TPDPLC problem must be minimized at a point in set D. It's evident that upper bound $ub_{i,j,s}$ is equal to lower bound $lb_{i,j,s+1}$. Let set $B_{i,j}$ be the bound set consisting of all upper bound and lower bound on arc (i, j). The number of elements is equal to $NB_{i,j}$+1 in set $B_{i,j}$ and kth component of set $B_{i,j}$ is $b_{i,j,k}$. The basic feasible solution of the TPDPLC problem can be denoted by $BF \in (F, L, U)$, where $F = \cup_k F_k$, $L = \cup_k L_k = \{l \mid l = lb_{i,j,s},\ s \in S_{i,j}\}$, $U = \cup_k U_k = \{u \mid u = ub_{i,j,s},\ s \in S_{i,j}\}$. Let bound set $B = L \cup U = \cup B_{i,j}$, then $BF \in (F, B)$. It means that the basic feasible solution of the TPDPLC problem has $m + n$ - 1 basic arcs and mn - ($m + n$ - 1) nonbasic arcs (lower bound or upper bound).

The form of the basic feasible solution for the TPDPLC problem is similar to that for the general transportation problems and the network flow problems. However, a distinguish feature of the basic feasible solution for the TPDPLC problem is that the bounds of the flow on any edge (u, v) is not fixed and can be any value in bound set $B_{u,v}$ (Bound of the flow on an edge for the general transportation problem and network flow problem can only one of two values, lower bound or upper bound. If the value of upper bound is unlimited, the flow bound is generally equal to zero). In our genetic algorithm, the problem is solved by the procedure called random bound selection procedure make using of the random search characteristic of genetic algorithm.

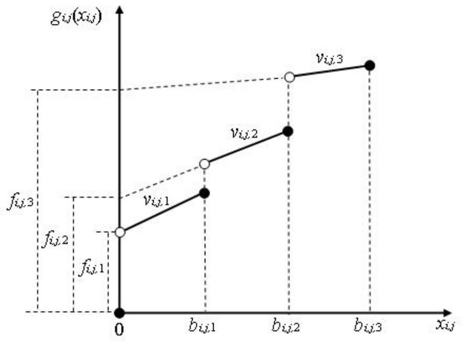

Fig. 1 Discontinuous piecewise linear cost function

3. Genetic algorithm

Genetic algorithm is a probabilistic algorithm that may converge at global optimal solution. The standard procedure of genetic algorithm can be found from Gen and Cheng (2000). In the rest of this section, we describe representation of candidate solution, initiation, crossover, and mutation operators in detail.

3.1. Sorted edge set encoding

One of the most important factors of a genetic algorithm is the encoding by which the chromosome is used to represent candidate solutions. The matrix encoding is a primitive representation for the transportation problem because the arc set of transportation network can be seen as a standard matrix. The paper [1] uses the matrix encoding to represent the candidate solution for the TPDPLC problem. However, the matrix encoding has not enough merits to evolution effectively and efficiently, so we hope to find another coding that is more compact and more effective to evolutionary computation for the TPDPLC problem.

Since the basic arcs of the basic feasible solution form a spanning tree of transportation network, the coding of spanning trees solution for the general transportation problems can be used for reference to encode the basic feasible solution for the TPDPLC problem. There is a variety of coding for spanning trees. They include link-and-node biasing [25], the preceding coding [26], prüfer numbers [12], edge sets [27], and so on. In these coding, edge sets coding, in which spanning trees are directly represented as sets of their edges, exhibits high locality that small changes in genotypes should correspond to small changes in solutions they represent and high heritability that solutions generated by crossover should combine features of their parents, and is proved to be a compact and effective coding for spanning tree. Moreover, edge sets coding can easily extend to represent the basic feasible solution for the TPDPLC problem. So we proposed a novel sorted edge set coding which not only inherit high locality and heritability of edge sets coding, but also extend edge sets coding to effective and efficient evolutionary computation for the TPDPLC problem.

In the sorted edge set coding, chromosome *c* representing the basic feasible solution for the TPDPLC problem consists of two parts: one part is sub-chromosome *f* which maps the basic edge set, and another part is sub-chromosome *z* which maps the positive nonbasic edge set. The positive nonbasic edge is edge whose flow is larger than zero. Representing only positive nonbasic edges in chromosome is more compact and more reasonable than representing all nonbasic edges in chromosome because most of nonbasic must be equal to zero to satisfy demand constraints and capacity constraints. Gene of chromosome is characterized by edge (s, d) (the flow amount on edge (s, d) can be added into gene, but is omitted for convenience of description). The number of gene is fixed to $m+n-1$ in sub-chromosome *f* through whole evolutionary process. The number of gene in sub-chromosome *z* is not a constant and can be increased or reduced while the basic feasible solution represented by chromosome varies.

All edges contained in chromosome *c* are sorted by a special rule. We assume spanning tree formed by the basic edges is $T=(V_T, E_T)$, of which root node is $r \in V_T$. Sub-chromosome *f* (one sorted edges set) can be attained through root first search that visit the node set V_T-r as deep as possible from neighbor to neighbor before backtracking. Sorting all positive nonbasic edges by the order created, we can obtain sub-chromosome *z* (another sorted edge set). Different chromosomes can be obtained if different node is selected to be root of spanning tree. We can randomly select a node to be the root in the initialization operator of individual. In the evolutionary operators, the father node of the first edge of sub-chromosome *f* is the root of the spanning tree represented by the current chromosome.

An important characteristic of sorted edge set with root first search pattern is the high efficiency of some basic operators on tree, for instance, it can be finished in $O(|E_T|)$ time to find a sub-tree whose root is any node $r \in V_T$ in tree T.

3.2. Appending Edge procedure

Creating spanning tree can be seen as a main procedure for creating new individuals. Positive nonbasic edges are produced during the creating spanning tree procedure. Moreove, *AppendEdge* procedure, which appends a new edge into the growing new individual, is a basic procedure in the initiation procedure and the crossover procedure. Let $G=(V,E)$ be the transportation graph, $T=(V_T,E_T)$ the growing tree where V_T is the growing node set and E_T is the growing edge set, P_T the set of node of whose demand (capacity) is not fully satisfied (exhausted), E_{UB} the nonbasic edge set, and $D(u)$ the residual amount of node *u*. *AppendEdge* procedure is below:

```
procedure AppendEdge ($V_T$, $E_T$, $P_T$, $E_{UB}$, $V$, $E$)
If $P_T \neq \phi$ Then
    Select randomly a node $u$ from the set $P_T$;
    If $\exists (u,v) \in E-E_{UB}$, and $v \in \{V-V_T\}$ Then
        Select randomly an edge $(u,v)$ from the set $E-E_{UB}$,
        and $v \in \{V-V_T\}$;
    Else
        Select randomly an edge $(u,v)$ from the set $E_{UB}$;
        $E_{UB} \leftarrow E_{UB}-(u,v)$;
        $D(u) \leftarrow D(u)+x(u,v)$;
        $D(v) \leftarrow D(v)+x(u,v)$;
    End If
Else
    Select randomly a node $u$ from the set $V_T$ where $\exists (u,v)$
    $\in E$, and $v \in \{V-V_T\}$;
End If
$x(u,v) \leftarrow \min(D(u),D(v))$;
$D(u) \leftarrow D(u)-x(u,v)$;
$D(v) \leftarrow D(v)-x(u,v)$;
$V_T \leftarrow V_T \cup \{v\}$;
$E_T \leftarrow E_T \cup \{(u,v)\}$;
If $D(u)=0$ and $u \in P_T$ Then $P_T \leftarrow P_T-\{u\}$; End If
If $D(v) \neq 0$ Then $P_T \leftarrow P_T \cup \{v\}$; End If
```

One *AppendEdge* procedure can not only add a new edge (u,v) into T, but also satisfy (exhaust) fully the demand (or the capacity) of node *u* or node *v*. In fact, the number of node in the set P_T is always 1 before the finishing last time application of *AppendEdge* procedures whether in the initiation procedure or in the crossover operator. So time of choosing one node from set P_T is thus constant. If a new edge $(u,v) \in E-E_{UB}$ can't be found, then an extra task is removing a corresponding edge from set E_{UB} (row 6~9 in *AppendEdge* procedures). Representing the graph G with adjacency list allows fast identification of the edges adjacent to each newly connected node, the identification operator's time is $O(|E|)$.

3.3. Initiation

As mentioned in section 3.1, the optimal solution of the TPDPLC problem need not be an extreme point of the convex set defined by constraints. It can occur within the convex set. However, we create only the extreme point solution for the initiation population, that is, the flow amount of each nonbasic arc is zero and sub-chromosome

$z = \phi$ in the initiation individuals. Since one *AppendEdge* procedure adds a new basic variable edge into *T*, |*V*|-1 times *AppendEdge* procedures can create a new individual satisfying all constraints from scratch. It is known easily that the edges of sub-chromosome *f* must be sorted by the pattern of root first search.

3.4. Crossover

Given the parents *P*1 and *P*2. Let *S*1 denote offspring individual will be created. To avoid complex repair operator, we developed following single point crossover operator based on sorted edge set representation.

Setp1: Choosing a random parent to be basic parent, for example, *P*1. A location k ($0<k<|E_{P1}|$) is randomly chosen in basic variables sub-chromosome *f* of *P*1. The edge pointed at by locate k is $e_k = (u_k, v_k)$;

Setp2: Visiting sequentially edges of sub-chromosome *f* (*P*1) of *P*1 from the location *k*, we can get a sub-tree $T(P1, k)$ of spanning tree denoted by sub-chromosome $f(P1)$. $T(P1, k) = (V_{T(P1,k)}, E_{T(P1,k)})$ where $|E_{T(P1,k)}| < |E_{P1}|$. The root of sub-tree $T(P1, k)$ is v_k;

Step3: Adding all nonbasic edge $(u, v) \in E_{UB}(P1)$ where $u \in V_{T(P1,k)} - \{v_k\}$ or $v \in V_{T(P1,k)} - \{v_k\}$ and $E_{UB}(P1)$ is equivalent to sub-chromosome $z(P1)$ into $E_{UB}(S1)$;

Step4: $|E_{P1}| - |E_{T(P1,k)}| - 1$ times *AppendEdge* procedures are used to append $|E_{P1}| - |E_{T(P1,k)}| - 1$ basic edges to sub-tree $T(P1,k) \cup \{e_k\}$. In the appending process, if *P*2 has a feasible edge $e = (u', v')$ where $u' \in P_{T(P1,k)}$, $e \in E_{P2} - E_{T(P1,k)}$, *AppendEdge* $(V_{T(P1,k)}, E_{T(P1,k)}, P_{T(P1,k)}, V_{P2}, E_{P2})$ is used to append feasible edge *e* to growing *T*. Otherwise, *AppendEdge* $(V_{T(P1,k)}, E_{T(P1,k)}, P_{T(P1,k)}, V, E)$ will be used to append feasible edge $e \in E - E_{P2} - E_{T(P1,k)}$ from graph *G*.

Demand (capacity) of every element is satisfied (exhausted) in set $V_{T(P1,k)}$ because $T(P1, k)$ is sub-tree of spanning tree *T*. So, the only parent node u_k of edge e_k is unsatisfied or unexhausted in all nodes of $T(P1, k) \cup \{e_k\}$. $|E_{P1}| - |E_{T(P1,k)}| - 1$ nodes must be inserted into sub-tree $T(P1, k) \cup \{e_k\}$ using $|E_{P1}| - |E_{T(P1,k)}| - 1$ times *AppendEdge* procedures, and a new individual can be created. Moreover, the edges of sub-tree $T(P1, k) \cup \{e_k\}$ must be sorted by the pattern of root-first search because all of these are inherit from parent individual *P*1. All edges of another sub-tree T_{sub} created by $|E_{P1}| - |E_{T(P1,k)}| - 1$ times *AppendEdge* procedures, of which root is node u_k, are also sorted by the same pattern. If all edges of sub-tree T_{sub} are appended to the edge set of sub-tree $T(P1, k) \cup \{e_k\}$, we can get a new offspring individual that all edges of sub-chromosome *f* are sorted by the root first search pattern.

3.5. Random pivot mutation

The network simplex method is an efficiently method for solving minimum cost flow problem. It pivots from the current basic feasible solution to a better adjacent one through adding an entering basic variable and removing a leaving basic variable. The idea that new solutions can be found by only a small change in the parent individual is suitable for the mutation operator of the genetic algorithm for solving the TPDPLC problem. Given parent individual *P* whose spanning tree is $T = (V_T, E_T)$, we choose randomly an element (i, j) from the nonbasic edge set $E - E_T$ as entering edge, instead of choosing a nonbasic edge which can improve the objective value at the fastest rate in the network simplex method. A circle *cl* is obtained after edge (i, j) is added to spanning tree *T*. However, it is not as simple as it might seem to choose a leaving edge $(r, s) \in cl$ because bound of the flow on each edge $(u, v) \in cl$ is not fixed and can be any value in bound set $B_{u,v}$ as mentioned in section 2. The following procedure is used to choose an edge to leave spanning tree *T*.

Step1: To entering edge $(i, j) \in cl$, we choose randomly the value of $k_{i,j}\max U_{i,j}$ where $k_{i,j} \in \{1, \cdots, NB_{i,j}\}$ and $k_{i,j}\max U_{i,j} \neq x(i,j)$. If $k_{i,j}\max U_{i,j} < x(i,j)$, then $k_{i,j}\max U_{i,j}$ is set to the lower bound $L(i, j)$ of edge (i, j). The difference $\triangle_{i,j} = x(i, j) - k_{i,j}\max U_{i,j}$ is used to estimate whether edge (i, j) should be the leaving edge. On the contrary, If $k_{i,j}\max U_{i,j} > x(i, j)$, then $k_{i,j}\max U_{i,j}$ is set to the upper bound $U(i, j)$ of edge (i, j). The difference $\triangle_{i,j} = k_{i,j}\max U_{i,j} - x(i, j)$ is used to estimate whether edge (i,j) should be the leaving edge;

Step2: If we number all edges in circle *cl* from edge (i, j), we can get edge set P_{odd} in which the index of each edge is odd and edge set P_{even} in which the index of each edge is even. To $\forall\ (r, s) \in P_{odd}$, if $k_{i,j}\max U_{i,j} < x(i, j)$, $k_{r,s}\max U_{r,s}$ is set to upper bound $U(r, s)$ where $k_{r,s} \in \{1, \cdots, NB_{r,s}\}$ and $k_{r,s}\max U_{r,s} > x(r,s)$; If $k_{i,j}\max U_{i,j} > x(i,j)$, $k_{r,s}\max U_{r,s}$ is set to lower bound $L(r, s)$ where $k_{i,j} \in \{1, \cdots, NB_{r,s}\}$ and $k_{r,s}\max U_{r,s} < x(r, s)$. On the contrary, to $\forall\ (r, s) \in P_{even}$, if $k_{i,j}\max U_{i,j} < x(i,j)$, $k_{r,s}\max U_{r,s}$ is set to lower bound $L(r, s)$ where $k_{r,s} \in \{1, \cdots, NB_{r,s}\}$ and $k_{r,s}\max U_{r,s} < x(r, s)$; If $k_{i,j}\max U_{i,j} > x(i, j)$, $k_{r,s}\max U_{r,s}$ is set to upper bound $U(r, s)$ where $k_{i,j} \in \{1, \cdots, NB_{r,s}\}$ and $k_{r,s}\max U_{r,s} > x(r, s)$;

Step3. Selecting edge $(r, s) \in cl$ so that $diff(r,s) = \min\{\min\{x(u, v) - L(u, v) \mid (u, v) \in cl$ and bound of the flow on edge (u, v) is the lower bound$\}$, $\min\{U(u,v) - x(u, v) \mid (u, v) \in cl$ and bound of the flow on edge (u, v) is the upper bound$\}\}$ where $diff(r, s)$ denotes the difference between $x(r, s)$ and bound of the flow on edge (r, s). If $(u, v) \in cl$ and bound of flow on edge (u, v) is lower bound, then $x(u, v) = x(u, v) - diff(r, s)$; If $(u, v) \in cl$ and bound of the flow on edge (u, v) is upper bound, then $x(u, v) = x(u, v) + diff(r, s)$;

Step4. When $(r,s) \neq (i,j)$, if $x(i, j) \in E_{UB}(P1)$, we remove edge (i, j) from $E_{UB}(P1)$; If $x(r,s) \neq 0$, we add edge (r, s) into $E_{UB}(P1)$.

The above pivot operator breaks the order of edges in individual P if $(r, s) \neq (i, j)$, so the edge set of T after pivoted need to reschedule in order to recover the normal order. The below procedure is used to implement the task.

Assume leaving edge (r, s) is the kth edge of the sorted edge set (sub-chromosome f) of individual P before pivoted, Visiting sequentially edges of T from the location k (not including edge (r, s)), we can get a sub-tree $T(T, k)=(V_{T(T,k)},E_{T(T,k)})$ of spanning tree T. Another sub-tree $T - T(T, k)$ can be obtained if sub-tree $T(T, k)$ is removed from spanning tree T. Sub-tree $T - T(T, k)$ and sub-tree $T(T, k)$ are connected through edge (i, j). Assume node i lies in the node set of sub-tree $T - T(T, k)$, and node j lies in the node set of sub-tree $T(T, k)$, next task will transfer sorted edge set $E_{T(T,k)}$ of $T(T, k)$ of which root is node s into sorted edge set $E'_{T(T,k)}$ of which root is node j. The detail procedure is below:

Visiting sequentially edges of set $E_{T(T,k)}$ from the edge of which son node is node j to edge e_{k+1}, we can obtain a track $l = (e_j, e_{j-1}, \cdots, e_z)$. To any two adjacent nodes e_x and e_{x+1} ($j \leq x \leq z-1$) in l, father node of edge e_x is son node of edge e_{x+1} and father node of edge e_z is node v_k. Then father node and son node of every edge in l are exchanged and all edges from edge e_j to e_{k+1} are insert into $E'_{T(P1,k)}$ while track l is obtained.

Let V_l be the node set of track l, the element number of set V_l is z-j+2. Any edge of which father node is one element of set V_l denotes a flag edge in edge set $E_{T(T,k)}$. Visiting each edges of set $E_{T(T,k)}$ from the last edge e_l to edge e_{j+1}, and inserting the segment that consist of edges from edge e_l to the first flag edge (including the first flag edge) and the segments that consist of edges between two flag edges (including the flag edge closer to edge e_{j+1}) into $E'_{T(P1,k)}$, we finish the rescheduling task.

An example of mutation operator is shown in Fig. 2. Sorted edge set of spanning tree T in parent individual P is {(1,2), (2,4), (4,8), (4,9), (2,5), (5,10), (1,3), (3,6), (3,7), (7,11), (11, 13), (7,12)} before the mutation is executed. Edge (10,13) and edge (1,3) are selected as the entering edge and the leaving edge respectively. Then, the sub-tree $T(T,7)$ = {(3,6), (3,7), (7,11), (11, 13), (7,12)} is rescheduled to $T'(T,7)$ = {(13,11), (11,7), (7,3), (3,6), (7,12)}. Appending all edge of $T'(T,7)$ and edge (10,13) into the T-$T(T,7)$, we can get the new individual T' whose sorted edge set is {(1,2), (2,4), (4,8), (4,9), (2,5), (5,10), (10,13), (13,11), (11,7), (7,3), (3,6), (7,12)}.

Finding sub-tree $T(T, k)$ can be finished in $O(|E_{T(T,k)}|)$ time as mention in section 3.1. Moreover, It can be found that the rescheduling sorted edge set of tree can also be finished in $O(|E_T|)$ time from above described procedure. So, the mutation operator is efficient.

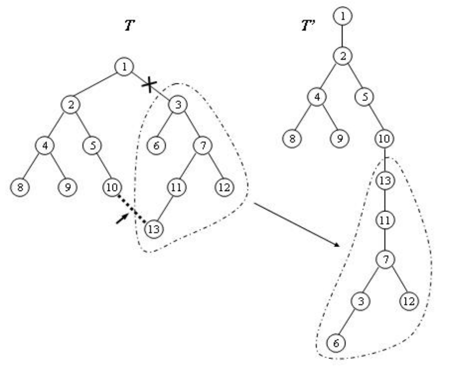

Fig. 2 spanning tree before and after mutated

4. Computational results

In the computation tests, three problem sizes, measured by $m \times n$, are used: 5×10, 10×10, 10×20. All the problems are fully connected. The total supply (demand) for three problem size are 5000, 10000, and 15000 respectively. Two kinds of experiments were distinguished. The first kind of experiment was performed in order to obtain influences of the number of segments in piecewise linear cost function on algorithm's performances. The test instances of each problem size were generated randomly. Within each test instance, four maximum numbers of segments were employed (2, 4, 6, 8) and other parameters were fixed. The net increased fixed costs between two adjacent segments were integers ranging 1200 through 2400. The variable unit costs on each segment were integers between 3~8.

Since a strong correspondence between the difficulty of an instance and the F/C ratio of the fixed costs to the variable costs of the optimal solution for the fixed charge transportation problem, we want also to know influences of the fixed cost on computational performance for the TPDPLC problem. It is the reason that we performed the second kind of computational experiment. Within each test instances, the maximum number of segments was fixed to 4. The variable unit cost on each segment was any integers between 3~8. The net increased fixed costs between two adjacent segments can be any integers from four different ranges: 100~400, 400~1600, 1600~6400, 6400~12800.

Our genetic algorithm and the matrix encoding genetic algorithm were coded in Java programming language. All computation tests were performed on the PC that the main frequency of CPU is Pentium (R)4 2.50GHz, operation system is Windows XP.

In our genetic algorithm, the population size was fixed at 100. The crossover rate and the mutation rate were 0.3 and 0.1 respectively. The pressure of tournament was 2. The generation limits was 2000. If the best individual of the next generation doesn't improve the current best solution, the worst individual of the next generation would be replaced by the current best solution. The evolutionary computing would be terminated if continuous 100 generation can not improve the current best solution. In the matrix encoding genetic algorithm, the crossover rate and the mutation rate were 0.05(c_1=0.35,c_2=0.65) and 0.2 respectively. The pressure of tournament was 2. The population sizes were 40. The generation limits was 10000.

In all test experiments, the solution quality difference between algorithm *A*1 and algorithm *A*2 was measured by *Diff_obj*(*A*1, *A*2) = $optimal_{A1}$ - $optimal_{A2}$ where $optimal_{A1}$ and $optimal_{A2}$ are the optimal solution obtained by algorithm *A*1 and algorithm *A*2 respectively. If *Diff_obj*(*A*1, *A*2) > 0, algorithm *A*1 can obtained better solution quality than algorithm *A*2 and vice versa. Similarly, the computational time differences between algorithm *A*1 and algorithm *A*2 were measured by *Diff_time*(*A*1, *A*2) = $time_{A1}$ - $time_{A2}$ where $time_{A1}$ and $time_{A2}$ are the computational time consumed by algorithm *A*1 and algorithm *A*2 respectively. If *Diff_ time*(*A*1, *A*2) > 0, algorithm *A*1 consumed more computational time than algorithm *A*2 and vice versa.

4.1. Results of the first kind of experiments

Let *SES-GA* and *MA-GA* denote our genetic algorithm based on sorted edge set coding and the matrix encoding genetic algorithm respectively. *Diff_obj*(*SES-GA*, *MA-GA*) for the first kind of computational experiments are showed in Fig. 3. Objective function values of each test problem are the average of 5 runs. *SES-GA* algorithm can obtained better solution qualities than *MA-GA* algorithm on each test problem because all *Diff_obj*(*SES-GA*, *MA-GA*) values are positive as shown in Fig. 3. The differences on solution qualities become larger while the number of segments increases. The tendency becomes more and more apparent if the problem sizes increase. In fact, the difficulty of an instance is in direct proportion to the number of segments for *SES-GA* whose genetic operators are relative to the number of segment. On the contrary, the genetic operators of *MA-GA* are not dependent on the number of segment. However, the fact shows that *MA-G* can not obtained better solution quality than *SES-GA* though the number of segments increases.

Other important information in Fig. 3 is that solution quality differences *Diff_obj*(*SES-GA*, *MA-GA*) become larger and larger while the size of problem increase. It proves that *SES-GA* is more suitable to solve the problems with large size than *MA-GA*.

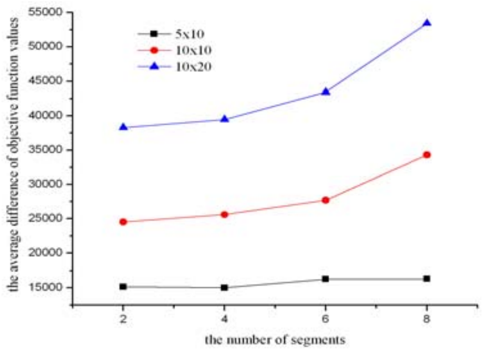

Fig. 3 average differences of solution quality

4.2. Results of the second kind of experiments

Diff_obj(*SES-GA*, *MA-GA*) and *Diff_time*(*MA-GA*, *SES-GA*) for the second kind of computational experiments are showed in Fig. 5 and Fig. 6 respectively. The solution quality and computation efficiency of *SES-GA* are both better than those of *MA-GA*. The strong tendencies that the larger fixed cost leads to the larger differences on the solution quality for problems with the same size and the same number of segment are shown in Fig. 5. Solution quality differences *Diff_obj*(*SES-GA*, *MA-GA*) and computational time differences *Diff_time*(*MA-GA*, *SES-GA*) are also larger and larger while the size of problem increase in the

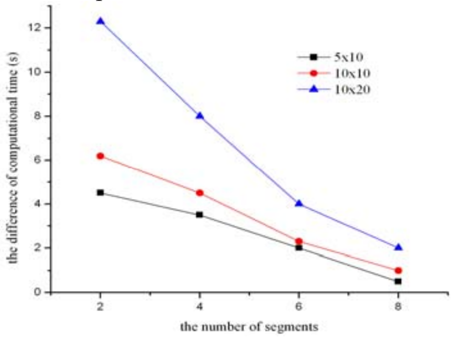

Fig. 4 differences of computational time

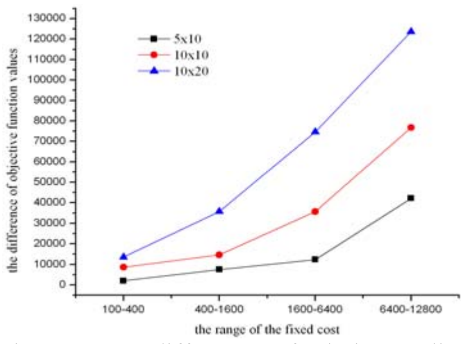

Fig. 5 average differences of solution quality second kind of experiments. However, it seem as if computational time difference *Diff_time*(*MA-GA*, *SES-GA*) are not sensitive to the values of the fixed cost for problems with the same size and the same number of segment, as shown in Fig. 6.

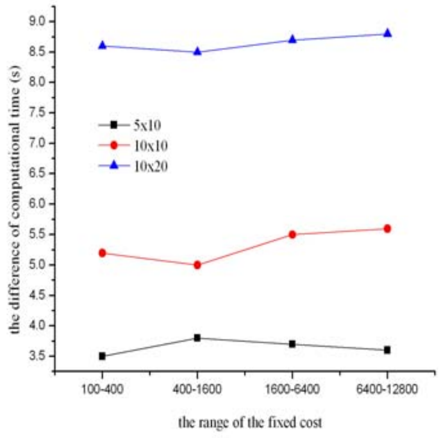

Fig. 6 differences of computational time

5. Conclusions

In this paper, we analyzed a transportation problem with discontinuous piecewise linear cost function and developed a genetic algorithm to solve it. Our genetic algorithm exhibits better optimization effect on solution quality and efficiency than the matrix encoding genetic algorithm. It should be the main reasons that our genetic algorithm utilizes the structure of spanning tree in the basic feasible solution and the possible values of the nonbasic variables are restricted to the flow bounds. The compact coding representing the basic feasible solution is another important factor.

In addition, the genetic algorithm developed in the paper can be applied to solve the fixed charge transportation problem as well as. It should be more effective and efficient because the flow bounds on edges are determinate.

Future research may concentrate on find some adaptive mechanisms to drive the evolutionary process away from local optima. It is also meaningful that extend our algorithm to other practical application areas, for example, the facility location and network design problems.

References

[1] Z Michalewicz,. and G.A. Vignaux, and M. Hobbs, “A nonstandard genetic algorithm for the nonlinear transportation problem,” ORSA J. Comput. vol.3, pp.307-316, 1991.

[2] J.M. Rousseau, “A cutting plane method for the fixed cost problem,” Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1973.

[3] K.G. Murty, “Solving the fixed charge problem by ranking extreme points,” Oper. Res., vol.16, pp.268-279, 1968.

[4] P.G. McKeown, “A vertex ranking procedure for solving the linear fixed charge Problem,” Oper. Res. vol.23, pp.1183-1191, 1975.

[5] U.S. Palekar, M.K. Karwan, and S. Zionts, “A branch and bound method for the fixed charge transportation problem,” Manage. Science, vol.36, pp.1092-1105, 1990.

[6] F. Ortega, and L.A. Wolsey, “A branch-and-cut algorithm for the single commodity, uncapacitated, fixed-charge network flow problem,” Networks, vol.41, pp.143-158, 2003.

[7] U.S. Shetty, “A relaxation decomposition algorithm for the fixed charge network problem,” Nav. Res. Log., vol.32, pp.327-340, 2003.

[8] D.I. Steinberg, “The fixed charge problem,” Nav. Res. Log., vol.7, pp.217-236, 1970.

[9] M. Sun, and P.G. McKeown, “Tabu search applied to the general fixed charge problems,” Ann. Oper. Res., vol.41, pp.405-420, 1993.

[10] M. Sun, J.E. Aronson, P.G. McKeown, and D. Drinka, “A tabu search heuristic procedure for the fixed charge transportation problem,” Eur. J. Oper. Res., vol.106, pp.441-456, 1998.

[11] J. Gottlieb, and L. Paulmann, “Genetic algorithms for the fixed charge transportation problem,” In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE Press, pp.330-335, 1998.

[12] Y. Li, M. Gen, and K. Ida, “Fixed charge transportation problem by spanning tree-based genetic algorithm,” Beijing Math., vol.4, pp.239-249, 1998.

[13] A. Balakrishnan, and S. Graves, “A Composite Algorithm for a Concave-Cost Network Flow Problem,” Networks, vol.19, pp.175-202, 1989.

[14] E.H. Aghezzaf and L.A. Wolsey, “Modeling Piecewise Linear Concave Costs in a Tree Partitioning Problem,” Discrete Appl. Math. vol.50, pp.101-109, 1994.

[15] R. Cominetti, and F. Ortega, “A Branch & Bound Method for Minimum Concave Cost Network Flows Based on Sensitivity Analysis,” Working paper, Departamento de Ingenieria Matematica, Universidad de Chile, Santiago, 1997.

[16] L. Chan, , A. Muriel and D. Simchi-Levi, “Supply Chain Management: Integrating Inventory and Transportation,” Working paper, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1997.

[17] K.L. Croxton, B. Gendron, and T.L. Magnanti, “Models and Methods for Merge-in-Transit Operations,” Trans. Science vol.37, pp.1-22, 2003.

[18] D. Kim and P.M. Pardalos, “A Dynamic Domain Contraction Algorithm for Nonconvex Piecewise Linear Network Flow Problems,” J. Global Optim., vol.17, pp.225-234, 2001.

[19] V. Gabrela, A. Knippelb, M. Minouxb, “Exact solution of multicommodity network optimization problems with general step cost functions,” Oper. Res. Lett. vol.25, pp.15-23, 1999.

[20] K. Holmberg, and J. Ling, “A Lagrangean Heuristic for the Facility Location Problem with Staircase Costs,” Eur. J. Oper. Res. vol.97, pp.3-74, 1997.

[21] K. Holmberg, “Solving the Staircase Cost Facility Location Problem with decomposition and Piecewise Linearization,” Eur. J. Oper. Res. vol.75, pp.41-61, 1994.
[22] A. Balakrishnan, T.L. Magnanti, and P. Mirchandani, “Network Design, Annotated Bibliographies in Combinatorial Optimization,” Edited by M. Dell'Amico, F. Ma_oli, and S. Martello (eds.), John Wiley & Sons, New York, NY, pp.311-334, 1997.
[23] T.G. rainic, A. Frangioni and B. Gendron, “undle-based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems,” screte Applied Mathematics, vol.112, pp.73-99, 2001.
[24] T.L. Magnanti, P. Mirchandani and R. Vachani, “Modeling and Solving the Two-Facility Capacitated Network Loading Problem,” Oper. Res. vol.43, pp.142-157, 1995.
[25] C.C. Palmer, and A. Kershenbaum, “Representing trees in genetic algorithms,” In: Proceedings of the First IEEE Conference on Evolutionary Computation, David Schaffer, Hans-Paul Schwefel, and David B. Fogel, Eds, IEEE Press, pp.379-384, 1994.
[26] F.N. Abuali, R.L. Wainwright, and D.A. Schoenefeld, “Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem,” In: Proceedings of the Sixth International Conference on Genetic Algorithms, Larry J. Eshelman, Ed, Morgan Kaufmann, pp.470-477, 1995.
[27] G.R. Raidl, and B.A. Julstrom, “Edge-sets: An effective evolutionary coding of spanning trees,” IEEE T. Evolut. Comput., vol.7, pp.225-239, 2003.
[28] W.M. Hirsch, and G. B. Dantzig, “The fixed charge problem,” Nav. Res. Log. vol.15, pp.413-424, 1968.

Su Sheng is a Ph.D. student at the School of Computer Science and Technology, Harbin Institute of Technology. He obtained his M.S. degree from Harbin Institute of Technology. His main research interests include CIMS, ERP, SCM, logistics, production planning and scheduling.

Zhan Dechen is a professor and Ph.D. supervisor at the School of Computer Science and Technology, Harbin Institute of Technology. He received his Ph.D. from Harbin Institute of Technology. His main research interests include CIMS, ERP, IDSS. He published over forty papers.

Xu Xiaofei is a professor and Ph.D. supervisor at the School of Computer Science and Technology, Harbin Institute of Technology. He received his Ph.D. from Harbin Institute of Technology. His main research interests include CIMS, ERP, IDSS. He published about one hundred and fifty papers.