The New Algorithm for Finding the Paths based on Coding Graph

Mingfu Wang
Dept. of Software Engineering , Shenzhen Polytechnic, Shenzhen, China
Email: wmingfu@21cn.com

***Abstract*—The most famous one for finding the shortest path is the Dijkstra, but it has some limitations. For example, when there are more than one shortest path between the source node and one special node, Dijkstra could find only one of them. Besides, the algorithm is quite complex. This article introduce a conception of coding graph, abstracting the problem of shortest paths and critical paths into the same mathematical model to describe and solve, presents a new algorithm of finding the paths. The algorithm extends first the data structure of the orthogonal list, so that the graph will be stored in the same storage space with the path searching process and result data. Codes for all nodes in the graph starting from the source node, using the rule of getting extremum in the weighing calculation and breadth-first, When accessing recursively the adjacent nodes from the current node, re-estimate the distance of neighboring node and entering edge list. if the distance of current node plus the weight of the edge to the neighboring node is less then (or greater than) the original distance of the neighboring node, then set this value as the new distance of this neighbor node, if the distance of current node plus the weight of the edge to the neighboring node is greater than (or less than) the original distance of the neighboring node, then the edge node will be deleted from the entering edge list, until all the nodes were coded and the coding graph of shortest path (or critical path) is created. For each node in the coding graph, starts searching from entry edge list recursively could get all shortest paths (or critical paths) and distances to the source node. Compared with the existing algorithms, this algorithm is simpler and more understandable, needs only 3n+5e storage unit that is much less then that of Dijkstra (which is n^2+2n). The time complexity O(n+e), which is also lower than the Dijkstra O(n^2).**

***Index Terms*—coding graph, shortest path, critical path, Dijkstra algorithm, extending orthogonal list, time complexity.**

I. INTRODUCTION

On the algorithm of shortest path and critical path, there were already many researches and applications. For example, represented as AOE(Activity On Edge Network), through adjusting and distributing the human and other physical resources to short the period of project, the key action which influences the project progress most. It is solving the problem of critical path of AOE network. But in network analysis, traffic plan, communication and computer science, great many problem could be formalized as finding shortest path model[1, 2], that is the problem of weighted graph shortest path of G =<V, E, W>. So, the researches on algorithm for finding shortest path attracted many professionals, and have extensive application area.

After the creative work of Dijkstra, many solutions for finding the shortest path have been presented [1~14]. Eli Olinick[3] and D.K.Smith[4] discussed it in 1966, but they are too simple, incomplete, and lacking time analysis, so have no much practicability. Cherkassky[12] give an theoretic analysis on seventeen algorithms for finding shortest path and give experimental evaluation. F. Benjamin Zhan tested fifteen of them, the result show that three of then are good, they are: TQQ (graph growth with two queues), DKA (the Dijkstra′s algorithm implemented with approximate buckets) and DKD（the Dijkstra′s algorithm implemented with double buckets）. The base of TQQ is graph growth theory, fit for computing the shortest path from one single node to all other node; the other two algorithm based on Dijkstra, fit better for computing the shortest path between two nodes[13]. Tan Guozhen[14] analyzed and evaluated more than twenty algorithms, including method of flag setting, modifying, dynamic planning, method based on linear algebra, method of elicitation and two-way elicitation, method based on neural network of fluid. All of these methods are based on Dijkstra, at the cost of time and space to get the improvement and extension, essentially not escape from the limitation of Dijkstra theory. So, the method simpler and more optimized than Dijkstra has not occurred yet.

Time–dependent model is more practical. Hongyan An and Guozhen Tan studied shortest path on time-varying network [15, 16] and got some achievements. But there is no mature algorithm occurred even today. It is still an area needs to do more study.

Traditional algorithm for finding shortest path[17] have shortage as follows: (1) needs to get the earliest and the latest time for every event occurred, calculate earliest and latest starting time for each action, estimate which is key action, the calculation is complex. (2) After the algorithm running, could get to know only which is and which is not key action, but could not get the number of critical path from the source node to the crossing-node. Neither could get each critical path. Fengsheng Xu[18]

Manuscript received July 1, 2010; revised September 16, 2010; accepted October 10, 2010.
Copyright credit, project number, Mingfu Wang.

doi:10.4304/jsw.7.1.1-8

improved the traditional algorithm, based on the extend-first searching, present a new algorithm. The good point is that need not to sort, but the shortage is using orthogonal list as storage structure, need three extend-first searches, and then could only get all the key actions, and not critical path. Fanzhen Meng[19] based on depth-first search, calculate all paths from source node to cross node, find the longest through analysis and comparison, so as to get critical path. Because backtracking recursively many times in the calculation, so it runs with low efficiency. Aiming at this shortage, Fang Liu[20] present a finding critical path algorithm that based on extend-first searching, using priority queue, and combining with dynamic planning principle. The time complexity decreased to O(n+e). However, because using adjacency list of graph, calculating the best value from bottom to top, and need to record some meaningful information, so it needs extra storage. Besides, in the same time, using priority queue according to entry-degree, need sort, where the time complexity was evaluated as O(n+e) which is not accurate. In addition, output all critical path needs transform information into two-dimensional array for each node, so it gets time at cost of space.

The author of the article breakthrough traditional idea about finding critical path algorithm, introduced the conception of coding graph at first time, present an new algorithm for finding critical path[21], which could find all the critical paths, key action, and their length from the source to crossing node.

This article simplified the conception of coding graph in [21], using an storage structure with extended orthogonal list, abstracting the shortest path and critical path as the same mathematic model, that is the model of path-finding. It aimed to transform the method of finding path into constructing a coding graph, present a new algorithm for finding path based on coding graph. In the same time, through presented path tree, transformed the problem of dynamic solving into the problem of modifying the code of path tree. Presented a dynamic algorism for finding shortest path when single edge weight varies.

The algorithm has the following feature:

(1) Set up a unified mathematic model of both the shortest path problem and problem of critical path, the algorithm is simple, and visual. The storage structure of coding graph with orthogonal list could be set up only by coding for the nodes of graph G and put deleted mark for some edge nodes of entry-edge list.

(2) Reverse search in the coding graph, all shortest path (or critical path) and their length could be found. The algorithm has time complexity O(n+e), optimal than that of Dijkstra (which is $O(n^2)$).

(3) No need for any additional condition on the graph, it is healthy and robustness. No additional conditions on the graph. The usual directed and undirected graph can be seen as a special case of weighted graph, so this algorithm is suitable for the general.

(4) No needs for additional temporary storage space, the graph, the temporary data of intermediate processing and the result data sharing same space. This algorithm requires only 3n+5e basic memory units, but Dijkstra algorithm requires (n^2 +2n).

(5) Take the maximum advantage of the graph formed before changing, by re-encoding on the changed shortest path tree (or critical path tree) it could be expanded to the shortest path (or critical path) of the fast dynamic algorithm.

II. BASIC CONCEPTS AND CHARACTERS

A. Definition of terms

Definition 1. For a given weighted graph G= < V, E, W >, set v_0, v_1, …, v_m ∈V , mark E(i, j)as directed edge from V_i to V_j, and E(0, 1) , E(1, 2), …, E(m-1, m)∈E, weight w_{01}, w_{12}, …, $w_{(m-1)m}$∈ W corresponding to each edge, sequence v_0 v_1 ... v_m as path connecting v_0 to v_m, $w_{01} + w_{12} + ... + w_{(m-1)m}$ as the length of the path. Among all paths connecting v_0 to v_m, the path with the minimum length is called shortest path, and the path with the maximal length is called critical path.

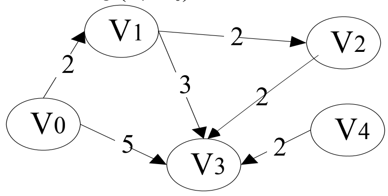
The usual undirected graph and directed graph can be viewed as a special case of the weighted graph, so the paper discusses only the weighted graph.

Definition 2. Set G =<V, E, W> as a weighted graph with n nodes, where node set V = {V_i | i=0, 1, 2, …, n-1}, edge set E={E(i, j) | i, j=0, 1, 2, …, n-1}, weight set for each edge corresponding to elements in E, W = {W_{ij} | i, j=0, 1, 2, …, n-1}, then the coding graph of shortest path in graph G corresponding source node V_0 (denoted by G (S, V0)) could be defined constructively as:

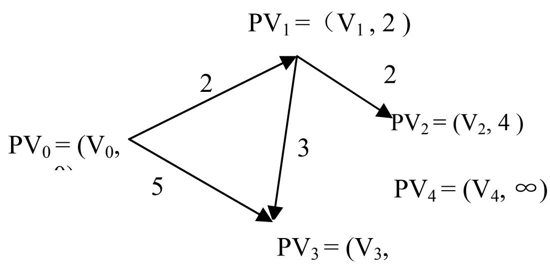
G(S, V_0)=<PV, E′, W′>

Where PV = {PV_i=(V_i, L_i)| i=0, 1, 2, …, n-1}, L_i as PV_i scalar quantity of nodes in G(S, V_0), its value is equal to the length of the shortest path between V_i and V_0 (if no path, take ∞, which will be called isolated node); E′⊆E, and E′={all edges of shortest path}, W′ is the set of weights corresponding to elements in E′, i.e, W′⊆W.

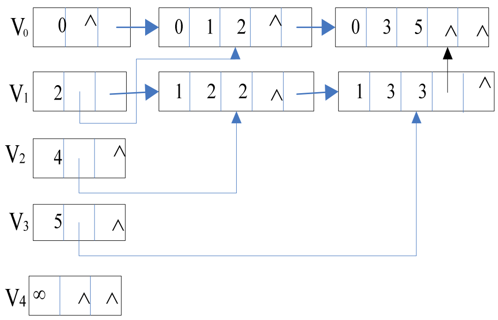
For example, Figure 1(B) is coding graph G(S, V_0) of G_1 corresponds to source node V_0 in Figure 1(A), while Figure 1(C) is the expansion of orthogonal list storage representations of G_1 (S, V_0).



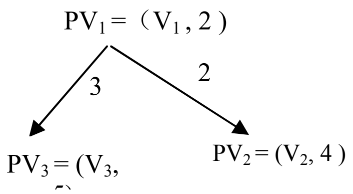
(A) weight graph G_1



(B) coding graph G(S, V_0) of G_1



（C）Expansion of orthogonal list representations of $G_1(S, V_0)$.



(D) shortest path tree $T(V_1)$

Figure 1. Schematic diagram G_1 and related concepts

Clearly, in the representation of expanded orthogonal list G_1（S, V_0）for each node V_i, search in opposite direction starting from entry-edge, until arrive the source node V_0, could find all shortest path from V_0 to V_i, and L_i in $PV_i = (V_i, L_i,)$ is the shortest path from V_0 to V_i .

Completely similar, let G =<V, E, W> is a n-node acyclic AOE network, in the above constructed definitions, if in $PV_i = (V_i, L_i)$ the node L_i is the longest path value from V_0 to V_i (if no path, take 0 value), then $G_1(S, V_0)$ is the coding graph to solve the critical path, same way, all the critical path, the key activities and length can be obtained from the source to either meeting-node.

Definition 3. Set G =<V, E, W> is a n-nodes weighted graph (or acyclic AOE network), $G' = G(S, V_0)$ is the shortest path code graph of G corresponds to source node V_0, in G' shortest path tree (or critical path tree) of node s is the biggest sub-graph $T(s)=<PV', E'', W''>$ of G', where s is called root-node of the tree, satisfying the following conditions:

(1) $PV' \subseteq PV$, $E'' \subseteq E'$, $W'' \subseteq W'$.

(2) For any $PV_i \in G'$, if the shortest path (or critical path) from V_i to V_0 pass through s, then $PV_i \in T(s)$ otherwise PV_i not belongs to T(s).

(3) There are path from node s to all other nodes in T(s), and further more, they are shortest path (or longest path). Especially, when $s=V_0$, $T(V_0)= G(S, V_0)$ - { isolated node }.

For example, Figure 1 (D) is the shortest path tree in G1 (S, V_0) on node V_1, shown in Figure 1 (B).

Definition 4. Set G = <V, E, W> is a n-nodes weighted graph (or acyclic AOE network), $G' = G(S, V_0)$ is the shortest path(or critical path tree) code graph of G corresponds to source node V_0, T (s) is the shortest path tree (or critical path tree), the smallest convex of T (s) is minimal sub-graph $T'(s)$ of G' with the following properties:

(1) $T(s) \subseteq T'(s) \subseteq G$.

(2) for any node $V_i \in G$ in G, but V_i not belongs to T(s), if in T(s) exist some one node V_k, make V_k as a directly next node of V_i, namely edge $E(i, k) \in E$, then $V_i \in T'(s)$, otherwise, V_i is not a node in $T'(s)$.

Specified, in graph G and code graph $G(S, V_0)$, the two node sets is obviously isomorphic (that is, $V \cong PV$), therefore, the node can be described in terms equivalently, for example, node PV_i and V_i is the same thing.

B. Characters

Theorem 1. Set G =<V, E, W> is a directed graph, without considering the order of edge nodes in the entry-edge list, the shortest path coding graph G(S, V0) = <PV, E, W> is unique.

Proof: the existence is evidently. Because in the graph G, any node V_i, if there is a shortest path from V_i to V_k, then according to the Dijkstra algorithm, you can find the shortest path and the length, suppose the path $(V_0, \ldots, V_f, V_i)$ is the shortest path, and length L, taking $PV_i = (V_i, L_i)=(V_i, L)$ edge $E(V_f, V_i)$ as the node of entry-edge list; if no path, take $(V_i, L_i)=(V_i, \infty)$, set table header field of the entry-edge empty.

For the unique, two ways to prove, first of all, the unique of L_i, guaranteed by the Dijkstra algorithm, the shortest path length is unique. The proof is as following: without considering the edge-node order of entry-edge list, entry-edge list is unique for any node V_i, that is, the set of edge-node is unique. In fact, suppose it is not unique, then exists at least two groups foreword-order set of nodes that are not completely the same. Say they are {Vp1} and {Vp2}, and $\{Vp1\} \neq \{Vp2\}$, set$Vj \in \{Vp2\}$, but Vj not belongs to {Vp1}, then foreword node set $V = Vj \cup \{Vp1\}$ is the preceding node set of V_i, this is contradict with the assumed condition that {Vp1} is of all the nodes V_i of the shortest path. So it is unique.

Theorem 2. Set G=<V, E, W> as a directed graph, in the coding graph G (S, V_0), L as a shortest path from V_0 to V_i, if L pass through the node V_k , and $L= L_1 \cup L_2$, then L_1 is the shortest path from V_0 to V_k, L_2 is the shortest path from V_k to V_i.

Proof: Without loss of generality, assume L_1 is not the shortest path from V_0 to V_k, but Q_1 is. Then, there exists at least one a different node V_t, and length of Q_1 is less then length of L_1, obviously $L' = Q_1 \cup L_2$ is a path from V_0 to V_i pass through V_k, the length of which is less then that of L, this is contradict with the assume that L is a shortest path from V_0 to V_i.

Theorem 3. In coding graph $G(S, V_0)$, search backward starting from the entry-edge list of PV_i, if $L_i = \infty$, then no path exists, otherwise, at the most through n-1（n is the number of nodes in graph G） steps could surely arrive the source node V_0, and the sequence of nodes obtained $V_i, V_{i1}, V_{i2}, \ldots, V_{im}, V_0$ is a shortest path from V_0 to V_i .

Proof: if the conclusion does not true, then in the sequence of searching path, there is at least one heavy node, such as V_k. In other words, exists at least one ring in the sequence of the search path. Assuming the node sequence ring is:

$V_i, V_k, V_f, \ldots, V_k, V_t, \ldots, V_0$

Obviously, the distance $\{L_f, L_t\}$ of the first two preceding nodes $\{V_f, V_t\}$ of the repeated nodes V_k, satisfying: $L_f+w_{fk}>L_t+w_{tk}$, according to the definition of coding graph, the length of node $V_k$could only be L_t+w_{tk}, thus, V_f could not be the preceding node of V_k, this is a contradict, so, at most through n-1 step could surely arrive the source node V_0. refferenced to the coding graph $G(S, V_0)$ and Theorem 2, the conclusion is proofed to be true.

Theorem 4. Set $G'=G(S, V_0)$ is the coding graph of G corresponds to source node V_0. $T(V_i)$ and $T(V_j)$ are the two shortest path tree of G', corresponds to node V_i and V_j. If $V_j \in T(V_i)$, then $T(V_j) \subseteq T(V_i)$. Otherwise, if $T(V_j)$ is a shortest path tree in $T(V_i)$, then $T(V_j)$ is also a shortest path tree of $G'= G(S, V_0)$.

Refferenced to the definition of shortest path tree, Theorem 4 be true.

Set $G =<V, E, W>$ as n-node acyclic AOE network, then in the critical path coding graph $G'=G(S, V_0)$, there are Theorem 1 ~ Theorem 4 similarly.

III. CODING GRAPH

A. Data structures

Orthogonal list of traditional graph stored method be extended to make as long as the node on the graph G to modify certain fields, delete some edge node of entry-edge list, it generates the expansion of orthogonal list storage of coding graph $G(S, V_0)$, the storage structure of graph G designed as follows:

(1) Node table header: header from the node to all the order structure (vector) is stored in order to randomly access any node in or out side list.

```
typedef struct Vertex //G (S, V0) node table structure
{  int L;                //the path length
   struct Ede *firstin;//entry-edge table header pointer
   struct Ede *firstout;//out-edge table header pointer
 }* VERTEX;
```

(2) Edge table: composed by the edges which representing adjacency relation between nodes in the graph.

```
typedef struct Ede  //structure of nodes in edge table
 {  int  tailvex;  //the subscript of starting node of edge
    int  headvex; //the subscript of ending node of edge
    int  weight;  //weight of edge
    struct Ede *headlink;//pointer of entry-edge table
    struct Ede  *taillink; //pointer of out-edge table
 }* PEDE;
```

B. Construction Algorithm

Construction algorithm of coding graph $G(S, V_0)$ (coding graph construction, CGC for simple) is a recursive algorithm, first of all to establish orthogonal list storage structure graph, introducing queue, on the graph G to breadth-first coding, establish distance values of the node, modify entry-edge list to generate the orthogonal list storage representation of shortest path (or critical path) coding graph.

Start from source node V_0, coding its adjacent node one by one, then starting from the adjacent coding sequence of their adjacent nodes, and to "first be encoded node's adjacent node" before "last encoded node's adjacent node" to be encoded, until the end of the recursive encoding, construction algorithm of shortest path (or critical path) encoding graph $G(S, V_0)$ described as follows:

(1) Initialization: Initialize the queue to be empty, set L_0 of the source node PV_0 to 0, for all other nodes initialize scalar L as ∞ (or L initialize 0).

(2) Take the source PV_0 into the queue.

(3) If the queue is not empty, then take out the first node PV_i, transfer L_i of node PV_i to the adjacent node, and take all the adjacent nodes with which the value of L has been modified into the queue. Delete non-shortest edge path (or critical path) edge node in the entry-edge list. Without loss of generality, assume current out-queue node is PV_i, and of PV_i the one-way adjacent nods are (V_{j1}, L_{j1}), (V_{j2}, L_{j2}), …, (V_{jk}, L_{jk}), and out edge are E (i, j1), E(i, j2), …, E(i, jk), the weight for corresponded adjacent edge are $w_{ij1}, w_{ij2}, w_{ij3}, \ldots, w_{ijk}$, for any adjacent nods（$V_{jm}, L_{jm}$）(m=1, 2, 3, …k）, make the mapping of transfer weight and take the minimum value(or maximum) ξ:

$$\xi(PV_i) = PV_{jm} = \begin{cases} (V_{jm}, L_i+w_{ijm}) & \text{if } L_{jm} > L_i + w_{ijm} \\ & (\text{or } L_{jm} < L_i + w_{ijm}) \\ (V_{jm}, L_{jm}) & \text{others} \end{cases}$$

and, when $L_{jm} < L_i+w_{ijm}$（or $L_{jm} > L_i+w_{ijm}$）, delete edge node E（i, jm）from entry edge list; when $L_{jm} > L_i+w_{ijm}$（or $L_{jm} < L_i + w_{ijm}$）, take the node PV_{jm} into queue.

(4) Repeat step 3 until the queue becomes empty.

C. Propertis

The paper discussed only the propertis of shortest path coding graph, of critical path coding graph is similar.

Theorem 5. Set $G' = G (S, V_0)$ as the coding graph of G, $T(V_j)$ is the shortest path tree corresponding with the node V_j in G'. Then, if the distance value L_j of nodes V_j changes, then the graph G, if and only if the new coding graph created from re-encoding the code graph $T(V_j)$ constructs the code graph of G. In other words, when and only when Access $T(V_j)$ changes the value of distance and entry edge list of the corresponding node, the new code graph of G after changing will be constructed.

Proof: According to the definition of coding graph and shortest path tree, it is obviously by recursively derive that all nodes of $T(V_j)$ need to re-encoding. If not, assume in graph G, there exists at least one node V_k not belongs to $T(V_j)$, of which the value of distance or entry edge list changed, then in $G' = G(S, V_0)$, the shortest path from V_k to source node must pass through V_j, according to definition of $T(V_j)$, V_k shall belongs to $T(V_j)$, that conflict with the assumption, therefore, only the node in $T(V_j)$ need to be re-encoded.

By Theorem 5 that, when distance value L_j of nodes V_j changes, need only re-encoding the shortest path tree $T(V_j)$ that rooted at node V_j, to construct the new coding graph. This make the existing graphs to be applied in the maximal scale, reduce time complexity. About re-encoding T (V_j) have the following conclusions.

Theorem 6. Set T (V_j) as the shortest path tree rooted at V_j in G' = G (S, V_0), then,

(1) If the distance value L'_j of V_j changed to be smaller (i.e, $L'_j<L_j<L_j$), then for the internal node T1={ PV| edge node including only the edge of T (V_j) in the entry edge list}, need only modify the distance value, that is for any $V_k \in T(V_j)$, change the distance value of node V_k into $L_k=L_k+L'_j-L_j$; While on the border node T2 = {PV | edge node including non-T(V_j) edge in entry edge list }, in addition to modify the distance value, but also remove all the non-T(V_j) edge nodes in the entry edge list. The resulting graph is the code graph of changed graph G.

(2) If the distance value L'_j changed bigger, (ie, $L'_j>L_j$), then starting from the node V_j, re-encoding recursively for T(V_j), we must consider the convex of T'(V_j), the resulting graph is the coding graph of changed G.

Proof: Without loss of generality, as long as proof the adjacent node, because the proof can be repeated recursively for all nodes. Suppose node V_k is the adjacent node of V_j, and both of them belong to T2, and E (j, k) and E (f, k) are two edges in the entry edge list of V_k, where E (f, k) does not belong to T (V_j) (that is, V_f is a node of G (S, V_0), but not T (V_j)), because the shortest path value L_k from V_0 to V_k via V_j is less than the original value L (the shortest path via node V_f), so, V_f is no longer the preceding node of V_k, E (f, k) must be removed from entry edge list. For node V_k on T1, from the definition of T (V_j) could know clearly that is enough only modify the distance value. Then according to Theorem 5, Theorem 6 (1) is proved to be true.

As for Theorem 6 (2), similarly if the adjacent node on V_j can be proved is enough. Suppose V_k is the adjacent node of V_j, if V_k has another one entry edge E (f, k) not belonging to T (V_j), and $V_f \in$convex T′(V_j), because the distance value L'_j of the tree's root node V_j getting bigger, so , when $L_f+w_{fk}<L'_j+w_{jk}$, must add the edge E (f, k) into the entry edge list of V_k , and remove E (j, k). In other words, when re-encoding T(V_j) recursively, it is necessary to consider the convex of T′(V_j), then according to Theorem 5, Theorem 6 (2) proved.

The dynamic shortest path algorithm could be simplified as code graph re-construction algorithm when single edge weight changed. From Theorem 6 could get the following Inference.

Inference 1. Set E(i, j) as a directed edge of code graph G(S, V_0) , if weight w'_{ij} changed to be bigger (i.e $w'_{ij}> w_{ij}$), then

(1) When E (i, j) is the only entry edge of node V_j, in G, then the distance value L′ of root node V_j of shortest path tree T (V_j) become bigger, the code graph constructed from re-encoding T (V_j) is the new graph of G after change.

(2) When E (i, j) is the only entry edge of node V_j in G (S, V_0), but there are other entry edges in G, such as node V_f, when $L_f+w_{fj}<L_i+w'_{ij}$, then delete the original entry edge list of V_j, make E (f, j) as a entry edge list edge node of V_j; when $L_f+w_{fj}=L_i+w'_{ij}$, add E(f, j) to the entry edge list of V_j; and then the graph constructed by processing the T(V_j) is the code graph of G after change.

(3) When the node V_j has more then one entry edges in G (S, V_0), as long as remove E(i, j) from the entry edge list of V_j, the result graph is the code graph of G after change.

Proof: Since E (i, j) is the only entry edge of node V_j in graph G, according to the definition of G(S, V_0), E (i, j) is also the only entry edge of V_j in code graph, therefore, will inevitably lead to the distance value of V_j in T(V_j) become bigger, according to Theorem 6(2), the inference 1(1) was true.

Base on the known conditions and the definition of coding graph, make the distance value of V_j become bigger, from Theorem 6(2), the inference 1(2) is true. From the definition of the coding graph, it is easy to know that the inference 1(3) is true.

Inference 2. Set E(i, j) as a directed edge in code graph G (S, V_0), if the weight w'_{ij} become smaller ($w'_{ij}<w_{ij}$), then

(1) When E (i, j) is the only entry edge of node V_j, then distance value L'_j of the root node V_j in the shortest path tree T(V_j) become smaller, the graph constructed from re-encoding only T(V_j) shall be the graph G after change.

(2) When the node V_j in the coding graph G(S, V_0) has more then one entry edges, then need only remove other entry edge nodes from it's entry edge list, leaving only this entry edge node, and the distance value L'_j of node V_j in the shortest path tree T(V_j) become smaller, and then the resulted graph by processing T(V_j) shall be the graph G（S, V_0）after graph G changed.

Proof: Since E (i, j) is the only entry edge of node V_j, and weight w'_{ij} become smaller, assume in G has a preceding node V_f, but $L_f+w_{fj}>L_i+w_{ij}>L_i+w'_{ij}$, therefore, E(f, j) could not be edge node of the entry edge list, so could only add the distance value L_i of node V_i with the new weight of E(i, j) to transfer to V_j($L_j = L_i+w'_{ij}$), i.e, the distance value of root node V_j of T(V_j) becomes smaller, according to Theorem 6 (1), Inference 2 (1) was true.

In the coding graph G (S, V_0), the node V_j has more than one entry edge, that means where are more than one shortest paths from V_0 to V_j, because the weight w_{ij} of E (i, j) become smaller, Therefore, the length value from V_0 via V_i to V_j become smaller, this path is the only shortest path from V_0 to V_j, according to the definition of coding graph, node V_j has only entry edge list for single edge node, so all other entry edge nodes must be removed from the entry edge list of V_j, retain only E(i, j) of which the weight value become smaller, then according to Theorem 6(1), inference 2(2) is true.

IV. PATH ALGORITHM

A. Static Path Algorithm

After construction of Coding graph G(S, V_0) it is easy to find all the shortest path (or critical path) and the length from the source node V_0 to other node V_k (k=1, 2,

…, n-1) in the graph G, obviously in coding graph G (S, V_0), if L_i in node PV_i =(V_i, L_i) is a non-0 finite value, then it is the length of the shortest path from V_i to the source node V_0. otherwise, no path exists. Thus, solving the shortest path (or critical path) algorithm, described as follows:

(1) Create the expanded representation of orthogonal list of graph G

(2) According to the shortest path (or critical path) construction algorithm for coding graph, construct the expanded representation of orthogonal list of coding graph G(S, V_0).

(3) In the expanded representation of orthogonal list of coding graph G(S, V_0), for any node PV_i, if the factor L_i is a non-0 finite value, then it is the shortest path (or critical path) length from source node V_0 to $V_{i.}$, while the tailvex value of entry-edge list node shall be the subscripts of the preceding node of V_i in the shortest path (or critical path). By recursive searching for each node in entry-edge could find all shortest path (or critical path). Otherwise, no path exists.

B. Dynamitc Path Algorithm

When the network environment changes, i.e, in the weighted graph G=<V, E, W> some of the edge weights changed, dynamic algorithm of the path can be reduced to the path algorithm of single weight change.

Obviously, the key of dynamic algorithm is the dynamic construction of coded graph, however, when the edge weights change, the removed edge node might re-enter into the list again, so, when constructing code graphs, if simply do delete operation, the deleted edge nodes need to be re-searched, resulting decreased efficiency of the algorithm. To solve this problem, add a flag field into the edge list structure, change the "delete" operation for the edge node from entry edge list into the operation "validate the delete flag", and the operation for the edge node "add into entry edge list" change to the operation "invalidate the delete flag" in the entry edge list. Therefore, in the constructing of coding graph, just do the operation of setting delete flag field into "on(validate)/off(invalidate)", remain the original entry edge list not changed.

This article only gives a dynamic shortest path algorithm, the critical path of the dynamic algorithm completely similar.

a) The expansion of storage structure

Expansion of edge list structure is as follows:

```
typedef struct Ede{
    int tailvex; //subscript of the edge starting point
    int headvex; //subscript of the edge ending point
    int weight; //weight of edge
    struct Ede * headlink; //entry edge list pointer field
    bool onof; //delete flag field, new added
    struct Ede * taillink; //out edge list pointer field
    } * PEDE;
```

b) The *dynamic shortest path algorithm*.

Set G (S, V_0) as a code graph of weighted graph G = (V, E, W) , weight value of the edge E (i, j) on node V_i, V_j changed from w_{ij} into w'_{ij}, according to the character of code graph, for single edge weighting change, shortest path algorithm is described as follows:

(1) If E (i, j) is not an edge of G (S, V_0), then turn to step 3.

(2) If w'_{ij} become bigger ($w'_{ij}>w_{ij}$), when E (i, j) is the only edge node in the entry edge list of node V_j , turn to step 6. Otherwise, the entry edge list of V_j, turn on deleted flag of E (i, j) in the entry edge list, and then turn to step 7.

If w'_{ij} become smaller($w'_{ij}<w_{ij}$), and in the entry edge list exists other edge nodes, turn on deleted flag for all other edge nodes, keep only for E (i, j) off. Turn to step 6.

(3) If w'_{ij} become bigger($w'_{ij}>w_{ij}$), then turn to step 8.

(4) if w'_{ij} become smaller （$w'_{ij}<w_{ij}$）, when $L_i+w'_{ij}>L_j$, turn to step 8, otherwise turn to step 5.

(5) If $L_i+w'_{ij}< L_j$, in the entry edge list of node V_j turn on deleted flag for all other edge node, turn off only the deleted flag for E (i, j), then turn to step 6. If $L_i+w'_{ij}= L_j$ then turn off the deleted flag for E(i, j) and turn to step 7.

(6) transfer the distance value L_i of node V_i and the weight value w'_{ij} of edge E(i, j) to node V_j, which is $L_j=L_i+w'_{ij}$; according to Theorem 6, calling CGC algorithm processing the distance value L and entry edge list on shortest path tree T(V_j), construct the coding graph G(S, V_0) with G changed.

(7)Starting from any code V_i in the graph G(S, V_0), began with entry edge list do reverse-search recursively to find all shortest paths and their lengths from source node to any node in the graph, then turn to Step 9.

(8)Coding graph G(S, V_0) did not change, no changes in the shortest path.

(9) algorithm end.

V. ALGORITHM ANALYSIS

A. The consumpion of storage space

Set G = (V, E, W) is a n-node directed weighted graph, Dijkstra algorithm adjacency matrix representation, need to provide auxiliary array dist [n], each element of the array includes two word Section: len field is distance from the source node V_0 to other, per field value is the order number from V_0 to the previous node; it needs a total of n^2+2n basic memory units. In this paper, graph G is stored with the expansion of orthogonal list, header array needs 3n storage units, edge table needs 5e (e is the number of edges in graph G) memory units, Total requirement is therefore 3n+5e basic memory units, apparently, in the dynamic algorithm when single edge weight changing requires 3n+6e basic memory units, apparently, save more storage space compared with traditional method.

B. Time Complexity

In the construction algorithm of coding graph G(S, V_0), each node in G get into the queue at most one time, so the number of nodes into the queue is not greater than n. When node V_i goes out of the queue, the internal cycling number of dealing with its neighboring nodes is equal to the out-degree d_i of node V_i. As the total time complexity accessing to all nodes of the adjacent node is $O(d_0+d_1+d_2+\ldots+d_{n-1})=O(e)$, so the time complexity of structural coding graph $G(S, V_0)$ is $O(n+e)$, superior to the traditional method of time complexity $O(n^2)$.

VI. ALGORITHM APPLICATION

Set $G_2=(V, E, W)$ as 9-node weighted directional graph, shown as Figure 2 (A) below. Without loss of generality, to solve the shortest path and the length from the node V_0 to other node, steps as following:

(1) Create the expansion of orthogonal list representation of G_2, shown as Figure 2(B). Field L be set value 0 in the table header (node V_0) or ∞ (other node).

(2) According to the construction algorithm of the shortest path coding graph, the orthogonal list representation of construction coding graph $G_2(S, V_0)$, shown as Figure 2(C), obviously is the sub-graph of $G_2(S, V_0)$ in Figure 2(B) that deleted E(1, 4), E(4, 7) and E(6, 8) from the entry-edge list of node V_4, V_7 and V_8.

(3) In the expansion of orthogonal list representation of shortest path coding graph $G_2(S, V_0)$, search from entry-edge of V_i (i=1, 2, …, 8), finish at the source node V_0, could find all shortest path from V_0 to all node V_i (i=1, 2, …, 8), information shown in Table 1. where has two shortest paths from V_0 to V_4 :$V_0 V_2 V_4$ and $V_0 V_3 V_4$, the length is 5; from V_0 to V_6 has two shortest paths: $V_0 V_2 V_4 V_6$ and $V_0 V_3 V_4 V_6$, with length 14.

VII. CONCLUSION

The article breakthrough traditional thinking of the solution path algorithm, by introducing the concept of coded graph, abstract the shortest path and critical path issues as the same mathematical model to describe and solve. Present a new path algorithm based on coding graph, and find the dynamic algorithm to solve shortest path when single edge weight changed. Compared with the existing shortest path algorithm is not only simple and intuitive, but also need storage space only 3n+5e(dynamic algorithm is 3n +6e) basic memory units, less then in traditional method (n2 +2n). Time complexity reduced to 0(n+e). That could efficiently, simply find the shortest path and length from any node to all other node in the graph, has a good adaptability.

In this paper, graph-based coding algorithm is the expansion of [21] for solving the critical path algorithm. More detailed study for the character of the coding graph, the expansion applied to time-dependent dynamic complex network shortest path algorithm, that is, the dynamic algorithm in complex case like add and remove nodes will be the subject of further study.

ACKNOWLEDGMENT

I would like to thank all those who helped in the preparation of this paper. In particular, I am grateful to Prof. HuJun Bao for his constructive suggestions. It is supported by the Open Project Program of the State Key Lab of CAD&CG(A1015), Zhejiang University.

REFERENCES

[1] A huja R K, M agnanti T L, Orlin J B. Network Flow s: Theory, Algorithm s and Applications. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[2] Donald M, Topkins. A K shortest path algorithm for adaptive routing in communication networks. IEEE Trans Communications, 1988, 36(7) : 855- 859.
[3] Eli Olinick [EB/ OL] . http://mail.informs.org/GROUP 96B/0299.html , 1996-06.
[4] Smith DK[EB/ OL] . http://mail.informs.org/GROUP 96B/0300.html , 1996-06.
[5] Mikkel Thorup. Floats, integers, and single source shortest paths. Journal of Algorithm s, 2000, 35(2) : 189- 201.
[6] Hanmao Sh i. Time work tradeoffs of the single source shortest paths problem. Journal of A lgo rithm s, 1999, 30(1) : 19-32.
[7] Friedhelm Meyer auf der Heide , Berthold vocking. Shortest path routing in arbitrary networks. Journal of Algorithms, 1999, 31 (1) : 105-131.
[8] Daniele Frigioni. Fully dynamic algorithms for maintaining shortest path trees. Journal of Algorithms, 2000, 34 (2) : 351-381.
[9] Donald Goldfarb. An*O*(*nm*) - time network simple algorithm for the shortest path problem. Operations Research, 1999, 47(3) : 445-448.
[10] Naesingh Deo, Chi-yin Pang. Shortest-path algorithms: taxonomy and annotation. Networks, 1984, 14 (2) : 275-323.
[11] Fengsheng Xu, new algorithm solving shortest path, computer project and science, 2006, 28(2):83-85.
[12] Cherkassky B V , Goldberg A V , Radzik T. Shortest paths algorithms: Theory and experimental evaluation. Mathematical Programming, 1996, 73(2) : 129-174.
[13] F B ZHAN. Three Fastest Shortest Path Algorithms on Real Road Networks[J] . Journal of Geographic Information and Decision Analysis , 1997, 1(1) : 69～82.
[14] Tan Guozhen. Shortest paths algorithms: Design, analysis, implementation and experimental evaluation. Department of Computer Science and Engineering, Dalian University of Technology: Technology Report 199901, 1999 (in Chinese).
[15] Hongyan An, dynamic algorithm of network shortest path, computer project and application, 2003.1:173-174.
[16] Guozhen Tan, shortest path algorithm in the time depending network, computer science, 2002.2:165-172.
[17] Weimin Yan, Weimin Wu, data structure, QingHa University publication company, 1997.
[18] Fengsheng Xu, Jing Huang. New algorithm solving critical path [J], computer application, 2004, 24(12):108-109.
[19] Fanzhen Meng. An algorithm solving critical path [J], computer project, 2001, 21(4):6-9.
[20] Fang Liu, Ling Wang, algorithm solving critical path based on dynamic planning theory, computer application, 2006, 26(6):1440-1442.
[21] Mingfu Wang, New Algprithm for Finding Critical Paths. Computer Engineering, 2008, 34(9):106-108.

Mingfu Wang born in 1956/12/25 at HuNan province of china, Associate professor of computer in Shenzhen Polytechnic. Achieved M.S. in mathematics from Zhejiang University in 1991. Research interests include computer graphics, volume visualization, computer vision, image processing, and applications on medical images.

Has been served as software engineer in Shenzhen Yuanwangcheng Multimedia computer Company, one of the project he directed, The multimedia colored painting system

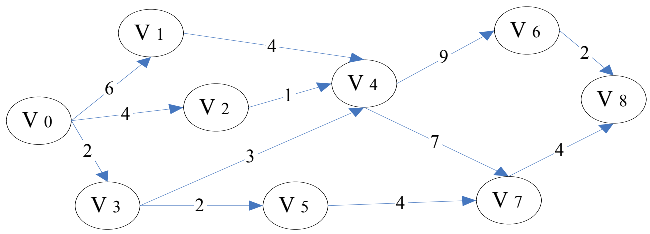
achieved inventive and creative Award by Technological Information System of Unit Nation (China Bureau) , now study instruct at shenzhen Polytechnic. Typical works includes:

[1] Mingfu Wang, Yong Zhou, Voxel-Coding for Surface Reconstruction from Contours, Computer Science, 2009, NO.7:34-38.

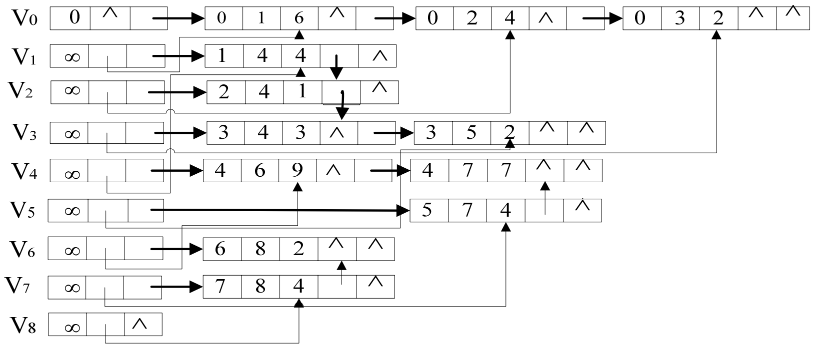
[2] Mingfu Wang, Zhiwen Qi., Research of Anti-copy and Plagiarism Monitoring System, ETCS 2009, Vol.2 :890-894.

[3] Mingfu Wang, Zhiwen Qi, Research and practice of Web server Optimization, ISECS 2009, 2009, Vol.2:432-436.

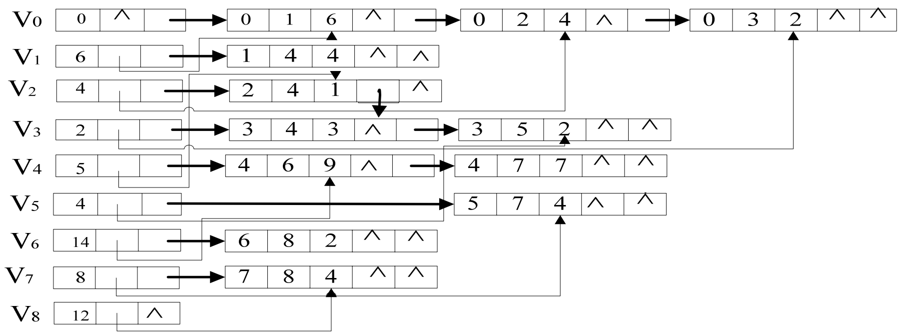
His research interests computer graphics and algorithm analysis.



(A) Weighted directed graph G_2



(B) Expansion of orthogonal list representation of G_2



(C) expansion of orthogonal list representation of coding graph $G_2(S, V_0)$

Figure 2. Expansion of orthogonal list representation of graph G_2 and coding graph $G_2(S, V_0)$.

Table 1. Information of coding Graph $G_2(S, V_0)$

Source node	Node V_i	Node of entry-edge list of V_i	Length of the shortest path from V_0 to V_i	Shortest path from V_0 to V_i
V_0	V_1	E(0, 1)	6	(V_0, V_1)
	V_2	E(0, 2)	4	(V_0, V_2)
	V_3	E(0, 3)	2	(V_0, V_3)
	V_4	E(2, 4), E(3, 4)	5	(V_0, V_2, V_4)和(V_0, V_3, V_4)
	V_5	E(3, 5)	4	(V_0, V_3, V_5)
	V_6	E(4, 6)	14	(V_0, V_2, V_4, V_6)和(V_0, V_3, V_4, V_6)
	V_7	E(5, 7)	8	(V_0, V_3, V_5, V_7)
	V_8	E(7, 8)	12	$(V_0, V_3, V_5, V_7, V_8)$