Finding all the negative cycles in a directed graph

Takeo Yamada[a, *], Harunobu Kinoshita[b]

[a] *Department of Computer Science, The National Defense Academy, Yokosuka, Kanagawa 239-8686, Japan*
[b] *Maritime Self-Defense Force, Maizuru, Kyoto 625-0087, Japan*

Abstract

Given a directed graph where edges are associated with weights which are not necessarily positive, we are concerned with the problem of finding all the elementary cycles with negative total weights. Algorithms to find all the elementary cycles, or to detect, if one exists, a negative cycle in such a graph are well explored. However, finding all the elementary cycles with negative cost appears to be unexplored. We develop an algorithm to do this based on the "divide-and-conquer" paradigm, and evaluate its performance on some numerical experiments. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Negative cycles; Enumeration; Directed graph

1. Introduction

Let $G = (V, E)$ be a directed graph [3] with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set $E = \{e_1, e_2, \ldots, e_m\} \subseteq V \times V$. A *path* is a set of edges of the form $\{(v^i, v^{i+1}) \in E \mid i = 0, 1, \ldots, k-1\}$, a *cycle* is a path with $v^k = v^0$, and a path is *elementary* if $v^i \neq v^j$ for all $i \neq j$. An elementary cycle is similarly defined. By $w(e)$ we denote an integer *weight* associated with edge $e \in E$. This may be positive or negative. For a path (or a cycle), its weight is defined to be the sum of the weights of constituent edges. An elementary cycle with negative weight is simply referred to as a *negative cycle*, and our problem is the following.

$\mathbf{P}_0$: List up all the negative cycles in G.

Researches have been done on algorithms to list up all the elementary cycles in directed as well as undirected graphs [4,9,11–14], but in these edge weights are not considered. Another related topic is the *negative cycle detection* [8], where the problem is to find a negative cycle, if one exists, or otherwise to prove nonexistence of such

---

* Corresponding author. Tel.: 81-468-41-3810; fax: 81-468-44-5911.
E-mail address: yamada@nda.ac.jp (T. Yamada).

a cycle. This problem is important in the *shortest path problem* [1] or the *minimum cost-to-time ratio cycle problem* [7], but in these finding only one negative cycle suffices.

These earlier works are related to our problem to some extent. For example, $\mathbf{P}_0$ is solved if we list up all the elementary cycles by any of the enumeration algorithms and pick up only those with negative weight. This is uninteresting, since usually graphs contain too many cycles for exhaustive enumeration. Even in such a case negative cycles may be relatively limited in number, and to our knowledge no researches have been done on the problem of enumerating only the negative cycles.

2. Framework of the enumeration algorithm

Given a directed graph $G = (V, E)$ with edge weight $w : E \rightarrow Z$, we can detect a negative cycle, if one exists, by modifying the *label correcting algorithm* [1] to solve the shortest path problem for graphs with nonnegative edge weights. Let $C = \{e^1, e^2, \ldots, e^k\}$ be such a negative cycle obtained. Then, we can divide $\mathbf{P}_0$ into the following set of subproblems $\mathbf{P}_i$, $i = 1, \ldots, k$.

$\mathbf{P}_i$: Find all the negative cycles of G that contain $e^1, e^2, \ldots, e^{i-1}$, but do not contain e^i.

More generally, for an elementary path $F = \{(v^i, v^{i+1}) | i = 0, 1, \ldots, k-1\}$ in G and a set of edges $R \subseteq E$ that is disjoint with F, an elementary cycle C is (F, R)-*admissible* if it contains all edges of F, but does not contain those of R. That is, C is (F, R)-admissible if and only if $F \subseteq C$, $R \cap C = \emptyset$. By $N(F, R)$ we denote the set of all (F, R)-admissible negative cycles. Here F and R are the sets of *fixed* and *restricted* edges, respectively. We pose the following problem.

$\mathbf{P}(\mathbf{F}, \mathbf{R})$: List up all the cycles of $N(F, R)$.

Clearly $\mathbf{P}_0$ is identical to $\mathbf{P}(\emptyset, \emptyset)$. Corresponding to $\mathbf{P}(\mathbf{F}, \mathbf{R})$ we introduce subgraph $G(F, R)$ as the graph obtained from G by removing vertices and edges incident to F. More precisely, $G(F, R)$ consists of the vertex set $V \setminus \{v^1, \ldots, v^{k-1}\}$ and the edge set $E \setminus R \setminus \{\bigcup_{i=0}^{k-1} E^-(v^i)\} \setminus \{\bigcup_{i=1}^{k} E^+(v^i)\}$, where $E^+(v)$ and $E^-(v)$ denote the sets of edges coming into and going out of v, respectively. The prototype algorithm $All\_NC_0(F, R)$ can be constructed in the *divide and conquer* [2,10] paradigm as the following recursive procedure, provided that we have an algorithm $An\_NC(F, R)$ that detects a negative cycle in $N(F, R)$ if one exists, and return $\emptyset$ otherwise.

Algorithm $All\_NC_0(F, R)$

Input: a simple path F, and $R \subseteq E$ such that $F \cap R = \emptyset$.
Output: $N(F, R)$.
Step 1: Call $An\_NC(F, R)$ to find $C = F \cup \{e^1, e^2, \ldots, e^k\} \in N(F, R)$.
　　If $C = \emptyset$ return. Otherwise output C.
Step 2: For $i = 0, 1, \ldots, k - 1$ do:
　　(i) Let $F_i := F \cup \{e^1, e^2, \ldots, e^i\}$ and $R_i := R \cup \{e^{i+1}\}$.
　　(ii) Call $All\_NC_0(F_i, R_i)$.

Unfortunately $An\_NC(F,R)$ is difficult to realize except for the case of $F = \emptyset$. To see this, for an elementary path F let its *initial* and *terminal* vertices be i_F and t_F respectively, and $\Pi(F,R)$ is the set of elementary paths form t_F to i_F in $G(F,R)$. For an arbitrary elementary path $\pi \in \Pi(F,R)$, $F \circ \pi$ denotes the elementary cycle obtained by connecting π to F. Clearly $F \circ \pi$ is (F,R)-admissible, and $F \circ \pi \in N(F,R)$ if and only if its weight satisfies

$$w(F \circ \pi) := w(F) + w(\pi) < 0. \tag{1}$$

By $\pi^\star(F,R)$ we denote the elementary path in $\Pi(F,R)$ of the minimum weight, and let

$$z^\star(F,R) := w(F \circ \pi^\star(F,R)). \tag{2}$$

Then, if $z^\star(F,R) < 0$ we have a negative cycle $F \circ \pi^\star(F,R) \in N(F,R)$, and otherwise $N(F,R) = \emptyset$. However, with negative edges in G finding $\pi^\star(F,R)$ is equivalent to the *longest elementary path problem* [6] which is $\mathcal{NP}$-hard in general.

Instead of calculating $z^\star(F,R)$ exactly, we evaluate its *lower* and *upper bounds*, $\underline{z}(F,R)$ and $\bar{z}(F,R)$. These bounds are discussed in the next section, but for the moment we assume these are available.

Then if

$$\underline{z}(F,R) \geqslant 0 \tag{3}$$

is satisfied, we conclude that $N(F,R) = \emptyset$, and subproblem $\mathbf{P(F,R)}$ is *terminated* as in Step 1 of All_NC_0. Or, if

$$\bar{z}(F,R) < 0, \tag{4}$$

we have a negative cycle $C = F \circ \bar{\pi}(F,R)$, where $\bar{\pi}(F,R)$ denotes a path in $\Pi(F,R)$ which is obtained together with $\bar{z}(F,R)$, as we shall see in Section 3. With C, we can generate subproblems of $\mathbf{P(F,R)}$ and call them recursively as in Step 2 of All_NC_0.

Otherwise, if

$$\underline{z}(F,R) < 0 \leqslant \bar{z}(F,R), \tag{5}$$

we neither have a negative cycle nor the evidence that $N(F,R) = \emptyset$. In such an *uncertain* case, with the set of edges $\{e^1, e^2, \ldots, e^k\} := E^-(t_F)$ in $G(F,R)$ emanating from t_F, we divide $\mathbf{P(F,R)}$ into the set of subproblems $P(F_i, R_i)$, $i = 1, \ldots, k$, where $F_i := F \cup \{e^i\}$ and $R_i := R$, and solve these problems by calling the algorithm recursively.

In the case of $F = \emptyset$ algorithms exist to solve An_NC as stated earlier, and thus no problem arises in All_NC_0. To sum up, we have the following.

Algorithm $All\_NC(F,R)$

Input: a simple path F, and $R \subseteq E$ such that $F \cap R = \emptyset$.
Output: $N(F,R)$.
Step 1: If $F \neq \emptyset$, go to Step 2. Otherwise let $C = An\_NC(\emptyset, R)$.
   If $C = \emptyset$ return; otherwise go to Step 4.
Step 2: Evaluate $\underline{z}(F,R)$. If $\underline{z}(F,R) \geqslant 0$, return.
Step 3: Evaluate $\bar{z}(F,R)$. If $\bar{z}(F,R) < 0$, put $C:=F \circ \bar{\pi} \in N(F,R)$ and
   go to Step 4. Otherwise, go to Step 5.
Step 4: (A negative cycle $C = F \cup \{e^1, e^2, \ldots, e^k\}$ is found)
   Output C. For $i = 0, 1, \ldots, k-1$ do:
    (i) Let $F_i:=F \cup \{e^1, e^2, \ldots, e^i\}$ and $R_i:=R \cup \{e^{i+1}\}$
    (ii) Call $All\_NC(F_i, R_i)$.
   return   ;
Step 5: (Uncertain case)
   Let $\{e^1, e^2, \ldots, e^k\}:=E^-(t_F)$ in $G(F,R)$.
   For $i = 1, 2, \ldots, k$ do:
    (i) Let $F_i:=F \cup \{e^i\}$ and $R_i:=R$.
    (ii) Call $All\_NC(F_i, R_i)$.

3. Lower and upper bounds

For an arbitrary subgraph $G' = (V', E')$ of G, let $d_j(v : v^0, G')$ denote the minimum weight of (not necessarily elementary) paths from v^0 to v in G' within j steps. This is also referred to as the *distance* of v, and satisfies the following recurrence relation [1,7]

$$d_j(v : v^0, G') = \min_{v' \in V'} \{d_{j-1}(v' : v^0, G') + w(v', v)\}, \quad \forall v \in V, \ j \geqslant 1, \qquad (6)$$

where $w(v', v)$ is the weight of $(v', v) \in E$, and

$$d_0(v : v^0, G) = \begin{cases} 0 & \text{if } v = v^0; \\ \infty & \text{otherwise.} \end{cases} \qquad (7)$$

Given an elementary path $F = \{(v^i, v^{i+1}) | i = 0, 1, \ldots, k-1\}$ in G, $d_{n-|F|}(i_F : t_F, G(F,R))$ gives the distance of i_F from t_F within $n - |F|$ steps. Let $\underline{\pi}(F,R)$ denote a (not necessarily elementary) path in $G(F,R)$ that attains the minimum distance, i.e.,

$$w(\underline{\pi}(F,R)) = d_{n-|F|}(i_F : t_F, G(F,R)). \qquad (8)$$

Such a path can be obtained as we calculate $d_j(v : t_F, G(F,R))$ via (6)–(7). Since elementary paths in $G(F,R)$ is at most of $n - |F|$ steps, $d_{n-|F|}(i_F : t_F, G(F,R))$ gives a lower bound to $w(\pi^\star(F,R))$, and

$$\underline{z}(F,R):=w(F) + w(\underline{\pi}(F,R)) \qquad (9)$$
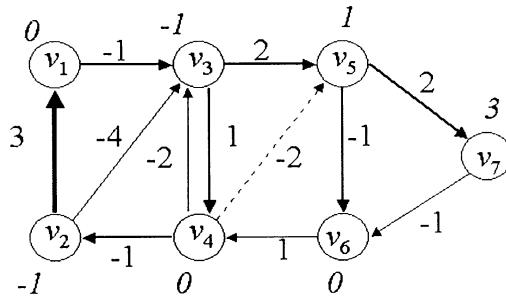
is a lower bound to $z^\star(F,R)$.
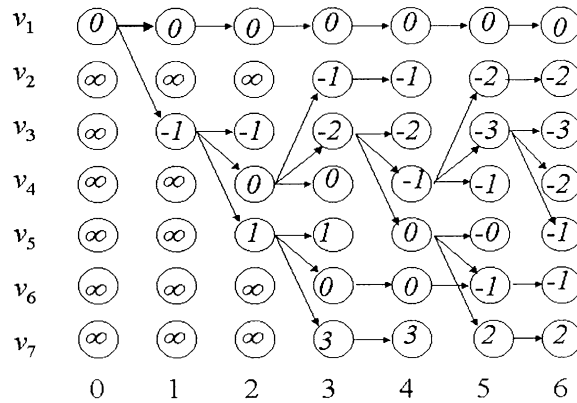
Fig. 1. Graph for Example 1.

Fig. 2. $d_j(v : v_1, F, R)$ for Fig. 1.

An upper bound can be found by applying the *Dijkstra's method* [5] to $G(F, R)$ with t_F as a starting vertex. Dijkstra's method usually finds shortest paths in a graph with nonnegative edge weights. If we apply the same algorithm to $G(F, R)$, for all $v \in V$ it finds an elementary path $\bar{\pi}(v : t_F, F, R)$ from t_F to v and its "distance" $\bar{d}(v : t_F, F, R)$. However, with negative edges in G, these may no longer be optimal. Nevertheless, since $\bar{\pi}(F, R) := \bar{\pi}(i_F : t_F, F, R)$ is elementary in $G(F, R)$, $\bar{d}(i_F : t_F, F, R)$ gives an upper bound to $w(\pi^\star(F, R))$, and thus the following is an upper bound to $z^\star(F, R)$.

$$\bar{z}(F, R) := w(F) + \bar{d}(i_F : t_F, F, R). \tag{10}$$

Example 1. Consider the graph of Fig. 1 with $F = \{(v_2, v_1)\}$ and $R = \{(v_4, v_5)\}$ shown in heavy and broken arrows, respectively. We have $w(F) = 3$, and applying the Dijkstra's method we obtain the tree shown in Fig. 1 in thick lines with $\bar{d}(v : v_1, F, R)$ given in italics at each $v \in V$. From this we have $\bar{\pi}(F, R) = v_1 v_3 v_4 v_2$ with $\bar{z}(F, R) = 2$. Fig. 2 shows $d_j(v : v_1, F, R)$, and from this we have $\underline{\pi}(F, R) = v_1 v_3 v_4 v_3 v_4 v_2$ with $\underline{z}(F, R) = 1$.

4. An illustrative example

Consider now the graph of Fig. 3, where e_k is shown by the number at each edge, and details are given in Table 1.   We start *All_NC* with $F=R=\emptyset$, which is subproblem **P_0**. Then, in Step 2, we get a negative cycle $C_1 = \{e_6, e_{14}, e_{11}, e_5\}$ as the output of *An_NC*. From this we generate 4 subproblems as shown in Table 2, where F and R describe each subproblem, and 'Cycle' denotes the output from *An_NC*. Here dashes (-) indicate $N(F,R) = \emptyset$, and question marks (?) are for the uncertain case of (5). Problem numbers will be explained soon.

By a recursive call with P_1 in Step 4, we obtain another negative cycle $C_2 = \{e_{21}, e_{30}, e_{39}, e_{26}, e_{23}, e_{19}\}$ as shown in Table 2, and from this generate 6 subproblems $P_2 \sim P_7$. These subproblems are all terminated due to condition (3). From P_8 we have $\underline{z}(F,R) = -68$ and $\bar{z}(F,R) = 50$, so we go to Step 5 of *All_NC* and generate 2 subproblems as follows. Table 3

Next, from P_{16} we obtain $C_3 = \{e_{21}, e_{30}, e_{39}, e_{26}, e_{23}, e_9, e_8, e_5, e_6, e_{14}\}$ and from this generate subproblems $P_{17} \sim P_{23}$ and P_{25} which are all terminated, except for P_{23}, due to (3) again. P_{26} is similarly terminated. In P_{23} we obtain $C_4 = \{e_{21}, e_{30}, e_{39}, e_{26}, e_{23}, e_{23}, e_9, e_4, e, 6, e_{14}\}$, which produces only one feasible subproblem P_{24}. This is terminated by (3), either.

All these processes are summarized in Fig. 4, where subproblems and their children are shown as a tree. Note that subproblems are generated and examined in the
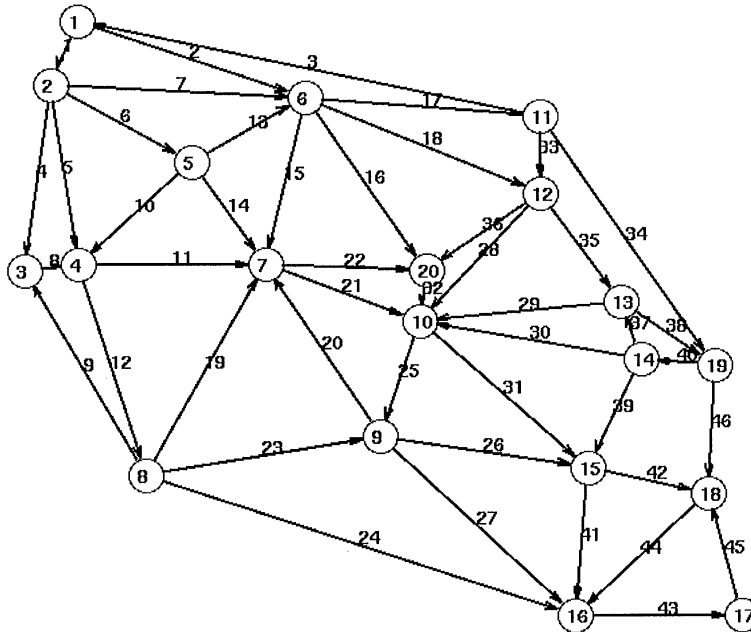

Fig. 3. Graph for the example of Section 4.

Table 1
Data for Fig. 3

Edge	Root	Top	Weight
e_1	v_1	v_2	53
e_2	v_1	v_6	180
e_3	v_1	v_{11}	353
e_4	v_3	v_2	−10
e_5	v_4	v_2	70
e_6	v_2	v_5	−30
e_7	v_6	v_2	−20
e_8	v_3	v_4	−104
e_9	v_8	v_3	183
e_{10}	v_5	v_4	10
e_{11}	v_7	v_4	−40
e_{12}	v_4	v_8	172
e_{13}	v_5	v_6	98
e_{14}	v_5	v_7	−20
e_{15}	v_6	v_7	133
e_{16}	v_6	v_{11}	175
e_{17}	v_6	v_{12}	190
e_{18}	v_6	v_{20}	162
e_{19}	v_8	v_7	10
e_{20}	v_9	v_7	159
e_{21}	v_7	v_{10}	30
e_{22}	v_7	v_{20}	120
e_{23}	v_9	v_8	−60
e_{24}	v_8	v_{16}	338
e_{25}	v_{10}	v_9	94
e_{26}	v_{15}	v_9	−90
e_{27}	v_9	v_{16}	201
e_{28}	v_{12}	v_{10}	134
e_{29}	v_{13}	v_{10}	150
e_{30}	v_{10}	v_{14}	−50
e_{31}	v_{10}	v_{15}	169
e_{32}	v_{20}	v_{10}	40
e_{33}	v_{11}	v_{12}	60
e_{34}	v_{11}	v_{19}	234
e_{35}	v_{12}	v_{13}	104
e_{36}	v_{12}	v_{20}	104
e_{37}	v_{14}	v_{13}	47
e_{38}	v_{13}	v_{19}	86
e_{39}	v_{14}	v_{15}	30
e_{40}	v_{19}	v_{14}	55
e_{41}	v_{15}	v_{16}	115
e_{42}	v_{15}	v_{18}	92
e_{43}	v_{16}	v_{17}	125
e_{44}	v_{18}	v_{16}	137
e_{45}	v_{17}	v_{18}	98
e_{46}	v_{19}	v_{18}	100

depth-first order. This is due to the recursive structure of *All_NC*. Shadowed circles represent the subproblems where negative cycles are found, and subproblems enclosed

Table 2
Subproblems generated from P_0

Subproblem	F	R	Cycle
P_1	$\emptyset$	$\{e_6\}$	C_2
P_8	$\{e_6\}$	$\{e_{14}\}$	?
P_{16}	$\{e_6, e_{14}\}$	$\{e_{11}\}$	C_3
P_{26}	$\{e_6, e_{14}, e_{11}\}$	$\{e_5\}$	—

Table 3
Subproblems generated from P_8

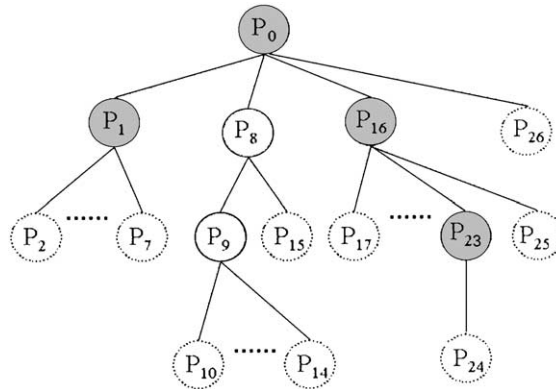| Subproblem	F	R	Cycle
P_9	$\{e_6, e_{13}\}$	$\{e_{14}\}$	?
P_{15}	$\{e_6, e_{10}\}$	$\{e_{14}\}$	—

Fig. 4. Tree of subproblems.

with broken lines are terminated due to condition (3). Thus, in this example *All_NC* generated 27 subproblem to find 4 negative cycles.

5. A stronger lower bound

In evaluating the lower bound $\underline{z}(F, R)$ we obtained the path $\underline{\pi}(F, R)$, but this may be non-elementary. If this happens to be elementary, we have $\underline{z}(F, R) = z^\star(F, R)$, and $An\_NC(F, R)$ is solved as explained previously.

Next, let us consider the case of non-elementary $\underline{\pi}(F, R)$, where the path meets to itself at some node c in $G(F, R)$. We divide $\Pi(F, R)$ into the set of elementary paths which go through c, and the set of those which do not go through c. First, let

$G^0(F,R,c)$ denote the subgraph of $G(F,R)$ obtained by removing vertex c and incident edges, and define

$$d^0(F,R,c):=d_{n-|F|}(i_F:t_F,G^0(F,R,c)). \tag{11}$$

Next, define subgraphs $G^1(F,R,c)$ and $G^2(F,R,c)$ as the subgraph of $G(F,R)$ obtained by removing $E^-(c)$ and $E^+(c)$ respectively. Let

$$d^1(F,R,c):=\min_{0<k<n-|F|}\{d^1_k(F,R,c)\}, \tag{12}$$

where

$$d^1_k(F,R,c):=d_k(c:t_F,G^1(F,R,c))+d_{n-|F|-k}(i_F:c,G^2(F,R,c)), \tag{13}$$

and put

$$\underline{z}'(F,R):=w(F)+\min\{d^0(F,R,c),d^1(F,R,c)\}. \tag{14}$$

Clearly this gives a lower bound to $z^\star(F,R)$ which is better than $\underline{z}(F,R)$, i.e.,

$$\underline{z}(F,R)\leqslant\underline{z}'(F,R)\leqslant z^\star(F,R). \tag{15}$$

Example 2. Consider the problem of Fig. 1 again with $F=\{(v_2,v_1)\}$ and $R=\{(v_4,v_5)\}$. The path $\underline{\pi}(F,R)$ meets to itself at $c=v_3$. Removing $E^-(c)$, no paths exist from $t_F=v_1$ to c and thus we have $d^0(F,R,c)=\infty$. Next, clearly $d_j(c:t_F,G^1(F,R,c))\equiv-1$ for $j\geqslant 1$ and

$$d_j(i_F:c,G^2(F,R,c))=\begin{cases}\infty, & j=0,1,\\ 0, & j=2,3,\\ -3, & 4\leqslant j\leqslant 6.\end{cases}$$

From these we obtain $\underline{z}'(F,R)=-4>\underline{z}(F,R)=-5$. Since $\bar{z}(F,R)=-4$ in Example 1, we conclude $z^\star(F,R)=-4$, and the path $\pi(F,R)$, shown in thick arrows in Fig. 1, is optimal.

In what follows, by All_NC' we denote the algorithm equipped with $\underline{z}'(F,R)$ instead of $\underline{z}(F,R)$ in All_NC.

6. Numerical experiments

To evaluate the performance of the developed algorithms, we have implemented All_NC and ALL_NC' in C language on an HP 9000 B132L workstation, and conducted a series of numerical experiments on the following test problems.

1. *Complete*: this is a directed complete graph K_n with vertices $\{1,2,\ldots,n\}$. Weights are given by

$$c(i,j)=\begin{cases}-1 & \text{if } |i-j|>\lceil n/p\rceil,\\ 1 & \text{otherwise,}\end{cases} \tag{16}$$

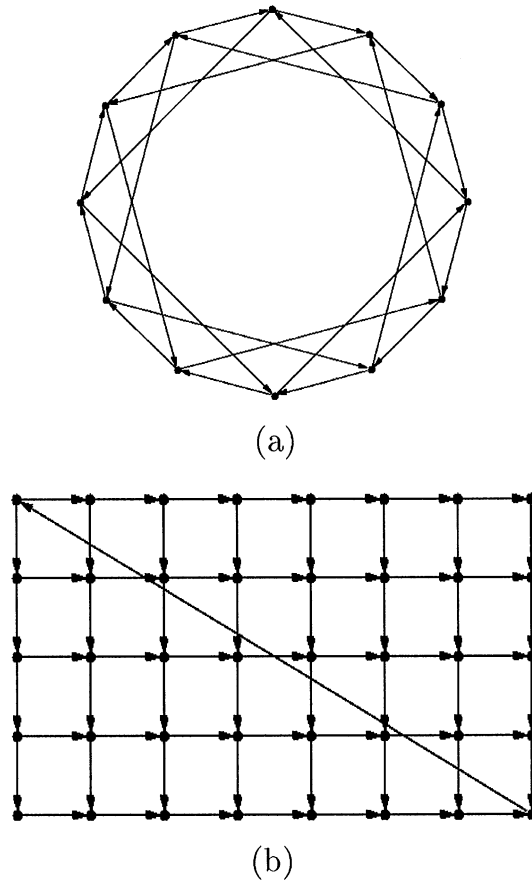where p is a parameter which is set to $p=1.4$ or $p=2.0$.

(a)

(b)

Fig. 5. Instances for numerical experiments: (a) $C_{12,1,3}$, and (b) $L_{5,8}$.

2. *Cord*: this graph $C_{n,K,p}$ consists of vertices $\{1,2,\ldots,n\}$ and edges $\{(i,i+1),\ldots,(i,i+K),(i,i-p)|i=1,\ldots,n\}$, where all additions and subtractions on vertex numbers should be done in **mod** n. All edge weights are 1, except for edges $\{(3k,3k-p)|1 \leqslant k \leqslant \lfloor n/3 \rfloor\}$, where weights are -1. Here K and p are parameters which are set to $K=1$ or $K=2$ and $p=3$ in our experiments.

3. *Lattice*: this is an $s \times t$ lattice $L_{s,t}$ with a feedback arc, consisting of vertices $\{v_{i,j} | i=1,2,\ldots,s, j=1,2,\ldots,t\}$ and edges $\{(v_{i,j}, v_{i,j+1}) | i=1,2,\ldots,s, j=1,2,\ldots,t-1\} \cup \{(v_{i,j}, v_{i+1,j}) | i=1,2,\ldots,s-1, j=1,2,\ldots,t\} \cup \{(v_{s,t}, v_{1,1})\}$. Edge weights are all 1, except for the bottom row edges where weights are -1.

Fig. 5(a) and (b) depict $C_{12,1,3}$ and $L_{5,8}$, respectively. We also implemented the Tarjan's algorithm [11] to list up all the elementary cycles for comparison. Tables 4–6 summarize the results of experiments for *Complete*, *Cord* and *Lattice* instances, respectively. Here shown are the number of negative cycles (#nc), the number of cycles found in Tarjan' algorithm (#cycle), the number of subproblems generated by

Table 4
Results of experiments for K_n

Graph	p	#nc	Tarjan		All_NC		All_NC$'$	
			#cycle	CPU	#prob	CPU	#prob	CPU
K_9	1.4	1	125664	5.4	3	0.0	3	0.1
K_{10}	1.4	1	—	—	3	0.0	3	0.1
K_{11}	1.4	23	—	—	54916	7.5	134	0.1
K_{12}	1.4	25	—	—	438542	91.5	158	0.1
K_{13}	1.4	27	—	—	3945802	637.6	184	0.1
K_{14}	1.4	1854	—	—	254037287	86803.9	4940262	1660.1
K_6	2.0	13	409	0.1	56	0.1	44	0.1
K_7	2.0	15	2365	0.2	112	0.2	58	0.1
K_8	2.0	246	16004	0.6	2069	0.3	894	0.1
K_9	2.0	364	125664	5.4	8706	1.9	1647	0.4
K_{10}	2.0	10348	—	—	162171	19.9	44412	6.4
K_{11}	2.0	19720	—	—	1073446	144.4	141655	27.5
K_{12}	2.0	699901	—	—	19758440	2816.0	4009789	796.7
K_{13}	2.0	1629217	—	—	175659211	53174.9	21343101	1342.4

Table 5
Results of experiments for $C_{n,K,3}$

n	K	#nc	Tarjan		All_NC		All_NC$'$	
			#cycle	CPU	#prob	CPU	#prob	CPU
10	1	0	49	0.00	1	0.00	1	0.00
20	1	4	877	0.07	22	0.01	22	0.01
30	1	71	25818	2.94	477	0.12	476	0.11
40	1	422	—	—	2341	0.63	2300	0.62
50	1	2790	—	—	18638	8.93	17855	7.48
60	1	41131	—	—	301472	193.17	300891	158.65
70	1	258222	—	—	1408320	771.02	1405706	660.35
10	2	2	416	0.02	8	0.00	8	0.00
20	2	12	50882	3.95	49	0.01	48	0.01
30	2	281	—	—	1766	0.54	1717	0.52
40	2	3775	—	—	17855	7.39	17148	6.92
50	2	44270	—	—	218141	127.73	209476	113.19
60	2	1230221	—	—	8017428	2616.40	7874398	1782.74

our algorithms (#prob), and CPU time in seconds. Dashes (—) indicate the cases where computation was terminated due to insufficient computer memories.

Tarjan was able to solve only a few smaller problems, mainly due to excessive memory requirements. For the case solved, all algorithms produced the same numbers of negative cycles. In *Complete* and *Cord* instances, *All_NC$'$* is superior to *All_NC* both in CPU time and memory requirements. However, for *Lattice* the difference between *All_NC* and *All_NC$'$* vanishes, because in this instance all cycles are necessarily elementary, and thus we have $\underline{z}'(F,R) \equiv \underline{z}(F,R)$.

Table 6
Results of experiments for $L_{5 \times t}$

Graph	#nc	Tarjan		All_NC		All_NC'	
		#cycle	CPU	#prob	CPU	#prob	CPU
$L_{5 \times 10}$	5	715	0.1	41	0.0	41	0.0
$L_{5 \times 20}$	210	3060	0.4	1107	0.2	1107	0.2
$L_{5 \times 30}$	1365	40920	12.5	7823	2.0	7823	2.0
$L_{5 \times 40}$	4845	—	—	31689	9.8	31689	9.8
$L_{5 \times 50}$	12650	—	—	94205	44.5	94205	44.5
$L_{5 \times 60}$	27405	—	—	229996	121.0	229996	121.0

7. Concluding remarks

We have developed an heuristic algorithm to list up all the negative cycles within the divide and conquer framework. Due to the gap between the lower and upper bounds for the shortest elementary path in graphs with negative edges, the algorithm often produces subproblems where existence of negative cycles is uncertain. By strengthening the lower bound, we were able to reduce this possibility, and with these algorithms solved problems with up to a few hundred vertices within reasonable CPU time.

For future researches we mention further strengthening of the lower and upper bounds. Recursion elimination from our algorithms may worth trying, since recursive algorithms are often inefficient in actual execution.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Englewood Cliffs, 1993.

[2] S. Baase, Computer Algorithms: Introduction to Design and Analysis, 2nd Edition, Addison-Wesley, Reading, MA, 1993.

[3] R.G. Busacker, T.L. Saaty, Finite Graphs and Networks: an Introduction with Applications, McGraw-Hill, New York, 1965.

[4] S. Chen, D.R. Ryan, A Comparison of three algorithms for finding fundamental cycles in a directed graph, Networks 11 (1981) 1–12.

[5] E. Dijkstra, A note on two problems in connection with graphs, Numer. Math. 1 (1959) 269–271.

[6] M.R. Garey, D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman and Company, San Francisco, 1979.

[7] R.M. Karp, A characterization of the minimum cycle mean in a digraph, Discrete Math. 23 (1978) 309–311.

[8] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Reinhart and Winston, New York, 1976.

[9] R.C. Read, R.E. Tarjan, Bounds on backtrack algorithms for listing cycles, paths, and spanning trees, Networks 5 (1975) 237–252.

[10] R. Sedgewick, Algorithms in C, 3rd Edition, Addison-Wesley, Reading, MA, 1998.

[11] R.E. Tarjan, Enumeration of elementary circuits of a directed graph, SIAM J. Comput. 2 (1974) 211–216.

[12] J.C. Tiernan, An efficient search algorithm to find the elementary circuits of a graph, Comm. ACM 13 (1970) 722–726.

[13] H. Weinblatt, A new search algorithm for finding the simple cycles of a finite directed graph, J. ACM 19 (1973) 43–56.

[14] J.T. Welch, A mechanical analysis of the cyclic structure of undirected linear graphs, J. ACM 13 (1966) 205–210.