情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-AL-125 No.7
2009/7/21

ビーコン配置問題と対偶問題に対する効率的近似アルゴリズム

汪　杰シュン[†1]　金　在成[†1]　佐々木　方太[†1]
趙　亮[†1]　永持　仁[†1]

ビーコン配置問題とは，与えられたグラフと整数 $L \geq 0$ に対して，つぎの条件を満たすような頂点集合の部分集合から最小のもの B を見つける問題である．ただし，任意の枝 e に対して，ある B の頂点が存在し，高々L 本の枝を使って e の（少なくとも一つの）端点に到達できる．この問題は，コンピュータネットワークにおけるリンクの観測から始まって，特に $L = 0$ の場合は，頂点被覆問題（Vertex Cover）と等価である．佐々木らは，Horton, Lopez-Ortiz (2003, $L = 1$ 相当) と Kumar, Kaur (2006, $L = 0$ 相当) の研究を任意の L に一般化させ，NP 困難性を示した上で厳密解法と欲張り法に基づいた似アルゴリズムを提案している（2008）．

本稿は，問題の拡張として**ロバスト被覆**を考える．すなわち，任意の枝 e について，高々L 本の枝を使って e の端点まで到達できる B の頂点数が，ある与えられた非負整数 r_e 以上でなければならない条件を付加する．我々は，この問題とそれの対偶問題に対し，効率的な近似アルゴリズムを与える．大規模コンピュータネットワークを用いて実験した結果，提案手法は従来法より実用的で精度も高いことがわかった．

Efficient Approximate Algorithms for the Beacon Placement and its Dual Problem

Jiexun Wang,[†1] Jaeseong Gim,[†1] Masahiro Sasaki,[†1]
Liang Zhao[†1] and Hiroshi Nagamochi[†1]

Given a graph and an integer $L \geq 0$, the *Beacon Placement Problem* (BPP) asks to find a minimum set B of nodes such that for all edges e, at least one of the two endpoints of e can be reached from some node (called an *L-beacon*) in B using at most L edges. In particular, it reduces to the Vertex Cover problem if $L = 0$. BPP arises from link-monitoring in computer networks. Generalizing the works of Horton and Lopez-Ortiz (2003, $L = 1$) and Kumar and Kaur (2006, $L = 0$), Sasaki, Zhao and Nagamochi (2008) formulated this problem and showed its NP-hardness for all L. They also provided an exact and an approximate algorithm.

In this paper, we first generalize the problem to a *robust covering* formulation which, for all edges e, asks that there exist at least r_e L-beacons that can reach at least one of the two endpoints of e, where $r_e \in \mathbb{Z}^+$ is a given robustness requirement. Then we propose efficient approximate algorithms for this and its dual problem. Studies on large-scale computer networks show the proposed algorithms are quite efficient and accurate in practice.

1. Introduction

Link-monitoring is a fundamental issue in computer network management (see 1), 2), 4)–7)). For that purpose, *active beacon based monitoring* is proposed, in which each link e is assigned to some node b (multiple links can be assigned to the same node), and b monitors the transfer delay of e by regularly sending probe packets to the two endpoints of e. The transfer delay of e is estimated by the difference of the two round-trip times of the probes. For accurate and secure monitoring, the assignment must be appropriate. For this, generalizing the works of Horton and Lopez-Ortiz[4], Kumar and Kaur[5], Sasaki, Zhao and Nagamochi[8] proposed the concept of *L-beacon*, which can monitor links reachable within L hops, i.e., at least one of the two endpoints of such a link can be reached using L or less links. In particular, a 0-beacon monitors all its incident links.

A natural question then asks, given a graph and an $L \geq 0$, to find a smallest set of L-beacons to monitor (i.e., cover) all links. This is called the *Beacon Placement Problem* (BPP). We note previous studies further assumed that, due to a feature of TCP/IP networks, a bridge (i.e., a link whose removal disconnects the graph) could be assigned to *any* beacon despite of the distance. In this paper, we simplify the formulation by removing that assumption. This does not change the nature of the problem. Then it is obvious that BPP reduces to Vertex Cover if $L = 0$. For any fixed $L \geq 1$, it reduces to Set Cover[8]. Therefore it is NP-hard for any fixed $L \geq 0$.

Since BPP is also a special case of Set Cover, we can apply the well-known greedy

†1 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University

© 2009 Information Processing Society of Japan

algorithm to get an $O(\log n)$-approximation, where n is the number of nodes. Sasaki et al[8] gave an exact algorithm and an efficient implementation of the greedy algorithm for BPP. However, they are not suitable for large-scale instances because of the long running time (see Section 4). In this paper, we first generalize the problem to a *robust covering* formulation that allows links be covered by at least r_e beacons for a given robustness requirement r_e. Then we give improved heuristics for this and its dual problem. For BPP, the new algorithm has $O(LR(m + n \log R))$ running time and requires $O(Rn + m)$ space, where m is the number of links (notice $m \geq n - 1$ for a connected graph) and $R = \max_e\{r_e\}$ is the maximum requirement. We note that in practice L and R are small, e.g., $L \leq 10$ and $R \leq 3$ are enough for the whole Internet. Thus the new algorithm is much more efficient than the previous ones. For the dual problem of BPP, our algorithm has $O(Lm)$ running time and requires $O(m)$ space. We remark that the constant factors are small too. Studies on large-scale instances show the new algorithms are quite efficient and accurate in practice for computer networks.

2. Formulation and Algorithm Sieve for BPP

Suppose we are given a connected graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges (i.e., links). Let $\mathrm{dist}(u, v)$ denote the u, v-distance in G, i.e., the number of edges on a shortest u, v-path. Let $B_d(v) = \{w \in V \mid \mathrm{dist}(v, w) \leq d\}$ denote the set of nodes that are reachable from a node $v \in V$ using d or less edges (d is called the *radius*). From the definition, we have $B_0(v) = \{v\}$ and $B_d(v) = \bigcup_{w\in\Gamma(v)} B_{d-1}(w)$, where $\Gamma(v)$ denotes the set of neighbors of v. We use $E_d(v) = \{e = (u, w) \in E \mid \{u, w\} \cap B_d(v) \neq \emptyset\}$ to denote the set of edges that can be reached from v using d or less edges.

Problem1 (BPP) Given $G = (V, E)$, an $L \geq 0$ and robust requirements $r_e \geq 0$ for all edges e, find a minimum set $B \subseteq V$ such that for all edges $e = (u, v)$, $|B \cap (B_L(u) \cup B_L(v))| \geq r_e$.

It can be written as the next Integer Programming.

$$\begin{aligned} \text{(IP) minimize} \quad & \sum_{i\in V} x_i \\ \text{s.t} \quad & \sum_{i\in B_L(u)\cup B_L(v)} x_i \;\geq\; r_e, \quad \forall e = (u, v) \in E \\ & x_i \in \{0, 1\}, \quad \forall i \in V. \end{aligned}$$

(Notice $B_L(v)$ is not part of the input.) Thus previous studies treated the case of $r_e \equiv 1$. Both of the exact and the approximate algorithms in 8) can be generalized to this formulation straightforwardly. In this paper, we give a faster Algorithm Sieve.

Algorithm Sieve consists of two phases. In Phase 1, we try to find a feasible solution, or stops with the conclusion that no one exists. In constructing the solution, we repeatedly check unchecked nodes v and let it be a beacon if there exists at least one unsaturated edge e in $E_L(v)$ (i.e., e is covered by at most $r_e - 1$ beacons). In that case, e is assigned to v. In Phase 2, we remove redundant beacons to get a minimal solution. This is done by checking beacons in the *reverse* order they were added.

Checking if there exists an unsaturated edge in $E_L(v)$ can be done by a BFS (Breadth-First Search) from v with depth L. Therefore we can implement Sieve in $O(mn)$ time (it is $O(m)$ for $L = 0$ since each edge is searched at most twice). This running time, however, can be $\Omega(mn)$ in general. To overcome this difficulty, we employ *coverage labels* to avoid useless searching.

Leaving the correctness proof to later, first we describe a subroutine BFS(v, L) to do BFS started from v with depth L, in which B holds the beacons found so far, k_e is the number of distinct beacons in B to which an edge e has been assigned. $B_c(v)$ holds the edges that will be assigned to v *if* v is added into B later. $E_c(v)$ is the set of unsaturated edges (i.e., $k_e < r_e$) in $B_c(v)$, and $E_r(v) = \{e \in E_c(v) \mid k_e = r_e - 1\}$. /* $\cdots$ */ are comments.

Subroutine BFS(v, L)

```
initialize a queue Q and enqueue(v), dist(v,v) = 0, mark v as searched;
while Q is not empty {
  u = dequeue();
  if dist(v,u) > L { return; } /* i.e., we have done */
  if L + 1 − dist(v,u) > ℓ_{u,i*} for i* = arg min_i{ℓ_{u,i}} {
```


© 2009 Information Processing Society of Japan

$\ell_{u,i^*} = L + 1 - \mathrm{dist}(v,u)$;
for all $e = (u,w) \in E$ **{**
$B_c(v) = B_c(v) \cup \{e\}$;
if $k_e < r_e$ **{** $E_c(v) = E_c(v) \cup \{e\}$; **}**
if $k_e == r_e - 1$ **{** $E_r(v) = E_r(v) \cup \{e\}$; **}**
if w has not been searched **{**
enqueue(w), $\mathrm{dist}(v,w) = \mathrm{dist}(v,u) + 1$, mark w as searched;
}
}
}
}

The main procedure of Sieve is as follows, where V' holds the unchecked nodes, and E' holds the unsaturated edges.

Algorithm Sieve

/ Phase 1: find a feasible solution */*
$B = \emptyset$, $V' = V$, $E' = E$, $k_e = 0$ for all $e \in E$;
while $E' \neq \emptyset$ **{**
if V' is empty **{ halt; }** (there is no feasible solution);
choose a node $v \in V'$ and let $V' = V' \backslash \{v\}$; */* see Remark */*
BFS(v, L**)**;
if $E_c(v) \neq \emptyset$ **{**
$B = B \cup \{v\}$, $k_e = k_e + 1$ for all $e \in B_c(v)$
$E' = E' - E_r(v)$; */* because all edges in $E_r(v)$ are now saturated */*
}
}
/ Phase 2: remove redundant beacons */*
$\ell_{v,i} = 0$ for all $v \in V$ and $i = 1, \ldots, R$; */* reset the coverage labels */*
for all $v \in B$ in the reverse order they were added **{**
$B = B \setminus \{v\}$, $k_e = k_e - 1$ for all $e \in B_c(v)$;
for all $e \in B_c(v)$ **{**
if $(k_e < M)$ **{**
BFS(v, L**)**;
$k_e = k_e + 1$ for all $e \in B_c(v)$, $B = B \cup \{v\}$;
break;
}
}
}
output B

Remark. The order for checking nodes is important. From our experience, the degree-decreasing order works well for computer networks.

Theorem1 The above algorithm Sieve can be implemented to have $O(LR(m + n \log R))$ running time and $O(Rn + m)$ space.

Proof. An edge is searched if and only if we can update some coverage label of one of its two endpoints. Since there are R labels for each vertex, and the maximum value is $L + 1$, we see each edge is searched at most $2R(L + 1)$ times. On the other hand, for each v, we can use a heap for finding $\mathrm{argmin}_i\{\ell_{u,i}\}$ and updating it. It is easy to see that other operations can be done in linear time. Therefore the total running time is $O(LR(m + n \log R))$. The space complexity is obviously $O(Rm)$. By remembering k_e at nodes, we can further reduce it to $O(Rn + m)$. ∎

Let us show the correctness of Algorithm Sieve.

Observation1 In Algorithm Sieve, it always holds that, for all v, there is no unsaturated edge in $E_{d-1}(v)$ for $d = \min_i\{\ell_{v,i}\}$, where we let $E_{-1}(v) = \emptyset$.

Proof. Omitted. ∎

Observation2 In BFS(v, L), there is no need to search nodes u satisfying $L + 1 - \mathrm{dist}(v,u) \leq \min_i\{\ell_{u,i}\}$.

Proof. Since the BFS is of depth L, we have $\mathrm{dist}(v,u) \leq L$. By $L + 1 - \mathrm{dist}(v,u) \leq \min_i\{\ell_{u,i}\}$, we see $d = \min_i\{\ell_{u,i}\} \geq 1$ and $\mathrm{dist}(v,u) \geq L + 1 - d$. Thus all nodes w that could be found by continuing BFS(v, L) from u must satisfy $\mathrm{dist}(u,w) =$

© 2009 Information Processing Society of Japan

$\mathrm{dist}(v,w) - \mathrm{dist}(v,u) \leq L - (L+1-d) = d-1$. In other words, $w \in B_{d-1}(u)$. On the other hand, by the previous observation, there is no unsaturated edge in $E_{d-1}(v)$. Hence there is no need to continue the BFS for u. ∎

Therefore we can have the next theorem.

Theorem2 Algorithm Sieve can correctly find a feasible solution for BPP or determine that there is no feasible solution. ∎

3. Dual Problem and Algorithm

Now let us consider a dual problem of BPP.

$$\begin{aligned} \text{(DP) maximize} \quad & \sum_{e\in E} r_e y_e \\ \text{s.t} \quad & \sum_{e\in E_L(i)} y_e \ \leq \ 1, \quad \forall i \in V \\ & y_e \in \{0,1\}, \quad \forall e \in E. \end{aligned}$$

For $L=0$ and $r_e \equiv 1$, this is nothing but the maximum matching problem, which can be solved in $O(\sqrt{n}m)$ time. In this paper, we give a fast heuristic. Again we can use the coverage label. In fact, since the coverage constraint is 1 this time, for each node v, only one label is enough.

Subroutine DualBFS(v, L)

initialize a queue Q and enqueue(v), $\mathrm{dist}(v,v)=0$, mark v as searched;
while Q is not empty **{**
  u = dequeue();
  if $\mathrm{dist}(v,u) > L$ **{ return; }** /* *i.e., we have done* */
  if $L+1-\mathrm{dist}(v,u) > \ell_u$ **{**
    $\ell_u = L+1-\mathrm{dist}(v,u)$;
    for all $e=(u,w) \in E$ **{**
      if w has not been searched **{**
        enqueue(w), $\mathrm{dist}(v,w) = \mathrm{dist}(v,u)+1$, mark w as searched;
      }
    }
  }
}

Algorithm DualSieve

$M = \emptyset$, $\ell_v = 0$ for all $v \in V$;
for all $e=(u,v) \in E$ **{**
  if $\ell_u \leq 1$ and $\ell_v \leq 1$ **{**
    DualBFS(u, $2L+1$), DualBFS(v, $2L+1$);
    $M = M \cup \{e\}$;
  }
}

In a similar way as Sieve for BPP, we can show the next theorem.

Theorem3 In $O(Lm)$ time and $O(m)$ space, Algorithm DualSieve can correctly find a feasible solution for (DP) or determine that there is no feasible solution. ∎

4. Experimental results

To evaluate the proposed algorithm, we studied a number of networks. All were tested on a PC with an Intel Xeon CPU X5260 (3.33GHz) and 16G RAM.

First we compared known algorithms using networks generated by GTgraph (`http://www.cc.gatech.edu/~kamesh/GTgraph/`). Based on the R-MAT model[3], they are supposed to be scale-free and small-world, which is considered good model for computer networks. The results are shown in Tables 1 and 2. For easy understanding, we also plot the data of Table 2 in a figure, where the data of LowerBound (Sasaki et al[8]) are omitted. We note that 10000-node is the limit for the exact algorithm in practice and our heuristics are fast and quite accurate.

For instances with more than 10,000 nodes, the CPLEX-based exact algorithm (with exponential running time) and the greedy algorithm (with $O(mn)$ running time) do not work (in the time limit of 3000 seconds). Therefore in the following we only show the result of Sieve and DualSieve. The next instance is also generated by GTgraph with

© 2009 Information Processing Society of Japan

表 1 頂点数 1,000，枝数 1,443 の GTgraph ネットワークに対する実験結果（$r_e \equiv 1$）．
Table 1 Results for 1,000-node, 1,443-edge GTgraph network with $r_e \equiv 1$.

	Exact[8]		Greedy[8]		LowerBound[8]		Sieve		DualSieve	
L	|B|	time (s)	|B|	time (s)	|B|	time (s)	|B|	time (s)	|B|	time (s)
0	460	0.01	474	0.00	458	0.00	472	0.00	437	0.00
1	163	0.09	183	0.01	162	0.02	178	0.00	152	0.00
2	70	0.15	78	0.00	70	0.04	80	0.00	64	0.00
3	38	0.14	44	0.02	38	0.05	43	0.00	37	0.00
4	21	0.25	26	0.02	21	0.10	24	0.00	20	0.00
5	13	0.44	15	0.06	12	0.15	15	0.00	10	0.00
6	8	0.47	12	0.07	8	0.20	11	0.00	8	0.00
7	6	0.44	7	0.09	6	0.22	8	0.00	6	0.00
8	4	0.41	6	0.11	4	0.25	5	0.00	4	0.00
9	4	0.42	5	0.13	4	0.28	4	0.00	4	0.00

表 2 頂点数 10,000，枝数 15,087 の GTgraph ネットワークに対する実験結果（$r_e \equiv 1$）．
Table 2 Results for 10,000-node, 15,087-edge GTgraph network with $r_e \equiv 1$.

	Exact[8]		Greedy[8]		LowerBound[8]		Sieve		DualSieve	
L	|B|	time (s)	|B|	time (s)	|B|	time (s)	|B|	time(s)	|B|	time(s)
0	4558	0.10	4745	0.01	4557	0.04	4701	0.00	4343	0.00
1	1534	75.2	1752	0.02	1531	0.78	1729	0.01	1396	0.00
2	667	177.26	824	0.07	662	3.09	799	0.00	589	0.00
3	307	1018.34	372	0.26	303	7.00	373	0.01	270	0.00
4	144	792.21	170	0.48	142	12.48	176	0.01	126	0.00
5	71	185.47	86	1.30	70	17.44	88	0.01	64	0.00
6	36	137.56	48	2.95	35	19.10	44	0.01	28	0.00
7	20	65.88	28	4.70	20	23.78	25	0.01	17	0.00
8	13	65.11	17	5.18	13	27.55	16	0.00	12	0.00
9	9	67.94	12	5.52	9	30.06	12	0.00	9	0.01

10,000,000 nodes. From the table, we see that the gap of Sieve and DualSieve for this instance is less than 2, which means the solution of Sieve (DualSieve) is within 2 times (at least half) of the optimal value.

Finally we tried a real network. The ITDK data is an Internet router network published by CAIDA (`http://www.caida.org/`), which has 192244 nodes and 607610 links. The results are shown in Table 4, in which we also show the results for $r_e \equiv 2$ and $r_e \equiv 3$. Notice that the result for (DP) with $r_e \equiv R$ is simply R times of the result with

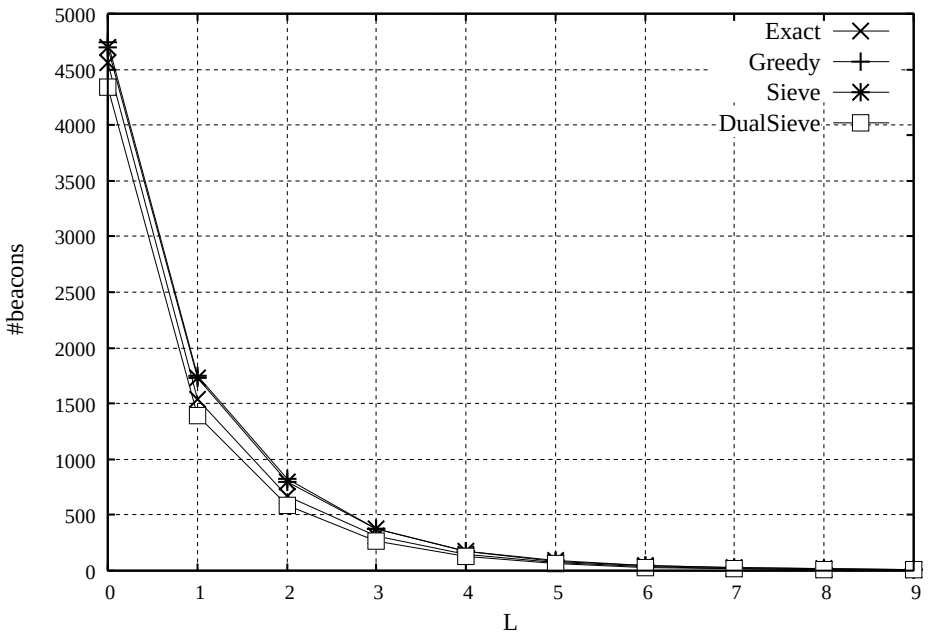

図 1 頂点数 10,000，枝数 15,087 の GTgraph ネットワークに対する実験結果（$r_e \equiv 1$）．
Fig. 1 Results for 10,000-node, 15087-edge GTgraph network with $r_e \equiv 1$.

$r_e \equiv 1$ (see the formulation). Again, we can see that both of Sieve and DualSieve are quite accurate.

5. Conclusion

In this paper, we formulated a robust version of the beacon placement problem (BPP) and showed efficient algorithms for this and its dual problem. Experimental results show they are fast (linear order in practice) and are quite accurate for large-scale computer networks.

Acknowledgment

This work is partially supported by the Grant-in-Aid for Scientific Research (no.

© 2009 Information Processing Society of Japan

表 3 頂点数 10,000,000，枝数 17,421,301 の GTgraph ネットワークに対する実験結果（$r_e \equiv 1$）．
Table 3 Results for 10,000,000-node, 17,421,301-edge GTgraph network with $r_e \equiv 1$.

	Sieve		DualSieve	
L	|B|	time (s)	|B|	time (s)
0	4593511	10.92	4244140	11.26
1	1529343	22.97	1230435	13.09
2	598350	33.56	454091	15.69
3	241308	45.01	174111	18.99
4	96997	57.80	66013	22.82
5	38368	71.50	24756	27.21
6	15005	85.11	9102	31.87
7	5890	95.70	3241	36.52
8	2269	103.32	1139	40.10
9	861	104.59	432	41.05

表 4 ITDK データに対する実験結果
Table 4 Results for the ITDK data

		Greedy		Sieve		DualSieve	
L	r_e	|B|	time (s)	|B|	time (s)	|B|	time (s)
0	1	75713	0.33	77978	0.19	69161	0.13
	2	n/a	n/a	190914	0.25	138322	0.13
	3	n/a	n/a	no solution	0.00	207483	0.13
1	1	18120	2.62	17918	0.32	15415	0.14
	2	n/a	n/a	38971	0.50	30830	0.14
	3	n/a	n/a	63737	0.67	46245	0.14
2	1	6569	33.30	6370	0.44	5552	0.17
	2	n/a	n/a	13480	0.73	11104	0.17
	3	n/a	n/a	21401	1.01	16656	0.17
3	1	2671	389.13	2596	0.56	2216	0.21
	2	n/a	n/a	5386	0.97	4432	0.21
	3	n/a	n/a	8394	1.36	6648	0.21
4	1	1128	2625.00	1112	0.68	932	0.26
	2	n/a	n/a	2290	1.22	1864	0.26
	3	n/a	n/a	3540	1.73	2796	0.26

20700010).

参 考 文 献

1) Bejerano Y. and Rastogi. R.: Robust Monitoring of Link Delays and Faults in IP Networks, *IEEE/ACM Trans. Networking*, Vol.14, No.5, pp.1092–1103 (2006).
2) Breitbart Y., Dragan F. and Gobjuka H.: Effective Network Monitoring, in Proc. ICCCN'04 (2004).
3) Chakrabarti. D., Zhan Y. and Faloutsos C.: R-MAT: A Recursive Model for Graph Mining, in Proc.SIAM Intl.Conf.on Data Mining (2004).
4) Horton J.D. and Lopez-Ortiz A.: On the number of distributed measurement points for network tomography, in Proc.ACM ICM'03, pp.204–209 (2003).
5) Kumar R. and Kaur J.: Practical Beacon Placement for Link Monitoring Using Network Tomography, in Proc.IEEE JSAC - SAMPLING 2006 (2006).
6) Moulierac J. and Molnar M.: Active Monitoring of Link Delays in Case of Asymmetric Routes, in Proc.IEEE ICNICONSMCL'06, pp.1–6 (2006).
7) Suha K., Guob Y., Kurosea J. and Towsley D.: Locating network monitors: Complexity, heuristics, and coverage, *Computer Communications* 29, pp. 1564–1577 (2006).
8) Sasaki M., Zhao L. and Nagamochi H.: Security-aware beacon based network monitoring, in Proc. IEEE ICCS 2008, pp.527–531 (2008).

© 2009 Information Processing Society of Japan