Chance-Constrained Programming Method of IT Risk Countermeasures for Social Consensus Making

Masaki Samejima, *Member, IEEE,* and Ryoichi Sasaki, *Life Member, IEEE*

***Abstract*—The authors address a social consensus making support in discussing countermeasures for information technology risks (IT risks). For supporting stakeholders' discussion on which IT risk countermeasures the stakeholders should implement, experts of the risk management estimate parameter values of the countermeasure, define a goal and constraints, and formulate the decision problem of the countermeasures to be implemented as one of 0–1 integer programming problems. Because parameter values and constraint values are uncertain, the decision problem is reformulated as a chance-constrained programming problem. The sample average approximation method is a well-known method for solving the chance-constrained programming problem. However, the computational time is still so long that the opinion leaders cannot use a solution of the chance-constrained programming problem in their discussion. The authors propose a high-speed chance-constrained programming method by aggregating the constraints that are generated by approximation of the problem in the sample average method. By applying the proposed method to real decision problems, the authors confirmed that computational time is decreased to 1 min while obtaining the same error rate and the same rate of the feasible solutions as a conventional method.**

***Index Terms*—Chance-constrained programming, constraint aggregation, information technology risk (IT risk), sample average approximation method, social consensus making.**

I. Introduction

INFORMATION technology (IT) risks are risks caused in information technologies that have become diverse due to the expansion and complexity of information systems [1], [2]. In order to reduce risk, system managers must implement countermeasures for IT risks. However, stakeholders of IT risks do not always agree to implement the countermeasures due to costs, loss of convenience, and so on. Without their agreement, some of the stakeholders are not willing to implement the countermeasures, which causes new IT risks. Some guidelines, such as "risk IT" [3], emphasize the importance of risk communication which is discussion the stakeholders to gain consensus. The authors are developing a new social-multiple risk communicator (MRC) [4] to support social consensus by numerous stakeholders.

Before the discussion with social-MRC, the experts who know much about IT risks estimate effects, cost, inconvenience, and the other parameter values of risk countermeasures. After that, the experts formulate the decision problem of the risk countermeasures to be implemented as a 0–1 integer programming problem [5] based on the estimated parameter values. Solutions of the problem are shown as candidates of the countermeasures that can be agreed upon by stakeholders. They choose one candidate combination of the countermeasures while considering public opinion. When the opinion leaders listen to public opinion (e.g., "effects of countermeasures should be changed"), they want to know the solutions in the case of changing the effects of countermeasures. However, it is difficult to determine a unique value for each parameter [6] because public requests on the parameter values are ambiguous and diverse. The authors have proposed the parameter value setting with random variables based on the public requests.

When the programming problem has parameters that are random variables, the programming problem is called a stochastic programming problem [7], [8]. Stochastic programming problems have several varieties of problems: chance-constrained programming problems [9], expected value optimization problems [10], entropy optimization problems [11], and so on. Because the stakeholders hope that the probabilities of violating the constraints are less than values decided by the stakeholders, the authors decide to reformulate the original 0–1 integer programming problem to a chance-constrained programming problem. Judgment on whether the chance constraints are satisfied requires many samplings for the random variables. It takes considerable time to solve the chance-constrained programming. Because the stakeholders have a few hours for discussion, it is expected that the chance-constrained programming problem is solved within a minute. The authors develop a chance-constrained programming method whose computational time is short enough to support the stakeholders' discussion in using social-MRC.

Manuscript received March 3, 2014; revised June 28, 2014; accepted September 9, 2014. Date of publication December 18, 2014; date of current version April 13, 2015. This work was supported in part by the Japanese Society for the Promotion of Science KAKENHI Grant 23300030. This paper was recommended by Associate Editor F.-Y. Wang.

M. Samejima is with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Suita 5650871, Japan (e-mail: samejima@ist.osaka-u.ac.jp).

R. Sasaki is with the School of Science and Technology for Future Life, Tokyo Denki University, Tokyo 1208551, Japan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2014.2376491

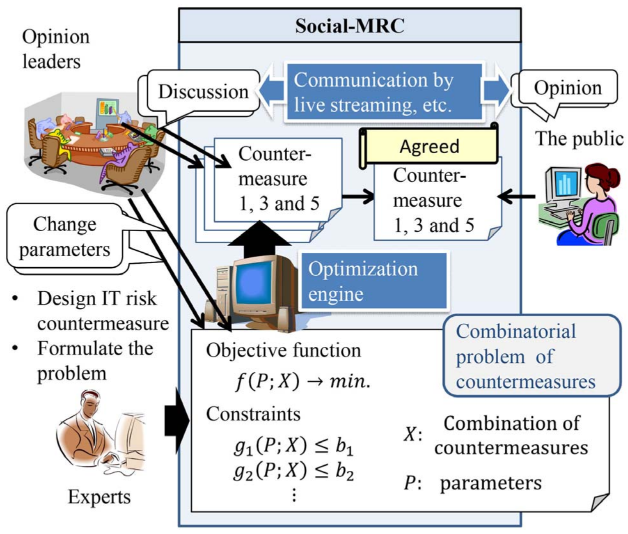


Fig. 1. Outline of social-MRC.

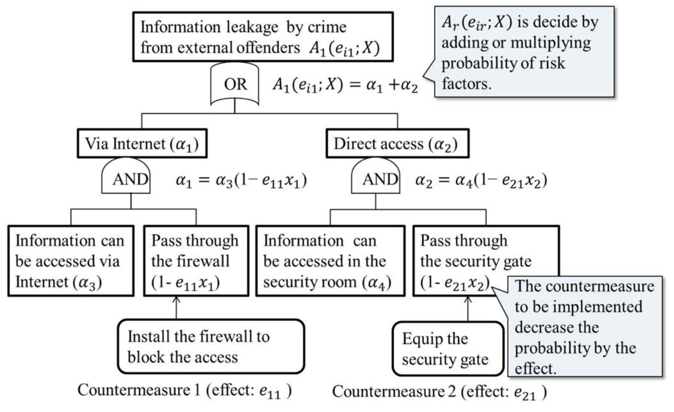

Information leakage by crime from external offenders $A_1(e_{i1};X)$
OR
$A_1(e_{i1};X) = \alpha_1 + \alpha_2$
$A_r(e_{ir};X)$ is decide by adding or multiplying probability of risk factors.
Via Internet (α_1)
Direct access (α_2)
AND
$\alpha_1 = \alpha_3(1 - e_{11}x_1)$
AND
$\alpha_2 = \alpha_4(1 - e_{21}x_2)$
Information can be accessed via Internet (α_3)
Pass through the firewall (1- $e_{11}x_1$)
Information can be accessed in the security room (α_4)
Pass through the security gate (1- $e_{21}x_2$)
The countermeasure to be implemented decrease the probability by the effect.
Install the firewall to block the access
Equip the security gate
Countermeasure 1 (effect: e_{11})
Countermeasure 2 (effect: e_{21})

Fig. 2. Estimation of occurrence probability $A_r(e_{ir}; X)$ by fault tree analysis.

II. Social-MRC

A. Outline of Social-MRC

It is difficult for a considerable number of stakeholders to attend a face-to-face discussion simultaneously to reach social consensus. The public selects several opinion leaders, and the opinion leaders reach a consensus. After the opinion leaders build a consensus through the mature discussion, the public judges whether they are for or against the consensus. For the opinion leaders' discussion, experts on IT risks define the purpose and constraints of the risk management, and provide information regarding the countermeasures to stakeholders. Fig. 1 shows the outline of social-MRC. Roles of experts, opinion leaders, and the public involved in social-MRC are as follows.

1) *Experts:* Experts design risk countermeasures to reduce the risk $r(r \in R)$ and estimate the effects of risk countermeasures, cost, inconvenience, and so on. Let X denote a 0–1 binary vector to express combinations of the risk countermeasures. A decision variable $x_i(1 \le i \le n)$ in the vector X indicates whether the risk countermeasure i is implemented $(x_i = 1)$ or not $(x_i = 0)$. And the programming problem to decide the combination of the countermeasures is formulated to the following 0–1 integer programming problem:

$$\text{Minimize } f(P; X)$$
$$\text{s.t. } g_j(P; X) \le b_j (j = 1, \ldots m)$$

where P is a vector of parameters, such as an effect $e_{ir}(\in P)$ of the ith countermeasure for the rth risk, and b_j is a limit value of each constraint. In general risk management, we should consider that the objective function $f(P; X)$ is the sum of expected loss $L_r A_r(e_{ir}; X)$ and the cost $c_i(\in P)$ of the ith countermeasure. The variable L_r is the loss of the rth risk and $A_r(e_{ir}; X)$ is an occurrence probability of the rth risk that is mitigated by the effect e_{ir}. The function $A_r(e_{ir}; X)$ is estimated by fault tree analysis [12], [13] as shown in Fig. 2

$$f(P; X) = \sum_{r \in R} L_r A_r(e_{ir}; X) + \sum_{i=1}^{n} c_i x_i. \tag{1}$$

Representative examples of constraints $g_j(P; X)$ are related to the sum of the occurrence probability $A_r(e_{ir}; X)$ and sum of cost c_i

$$g_1(P;\ X) = \sum_{r \in R} A_r(e_{ir};\ X) \le b_1 \tag{2}$$

$$g_2(P;\ X) = \sum_{i=1}^{n} c_i x_i \le b_2. \tag{3}$$

2) *Opinion Leader:* Each opinion leader changes the parameter values based on opinions from the public and solves the problem to obtain a solution that can be agreed upon by each opinion leader and the public (e.g., risk-sensitive opinion leaders often set small values to the effect of the countermeasure and obtain the solution under assumption of pessimistic situations). Then opinion leaders support different solutions depending on different parameter values.

3) *The Public:* The public chooses an opinion leader based on the content of the discussion, and states opinions about the formulation of the problem, the solutions supported by opinion leaders, and the parameters.

In order to support the discussion among the stakeholders, social-MRC has an optimization engine to solve the 0–1 integer programming problem quickly. The public can watch the discussion via live streaming and send in their opinions via microblog.

B. Chance-Constrained Programming of IT Risk Countermeasures

As the authors stated in Section II-A, opinion leaders take the public opinion into consideration when setting parameter values, and solve 0–1 integer programming problems repeatedly. Then the original 0–1 integer programming problem changes as shown in Fig. 3. Opinions from the public are qualitatively expressed, such as "effects of risk countermeasures are overestimated" and "we have to consider increasing cost," which make it difficult to determine

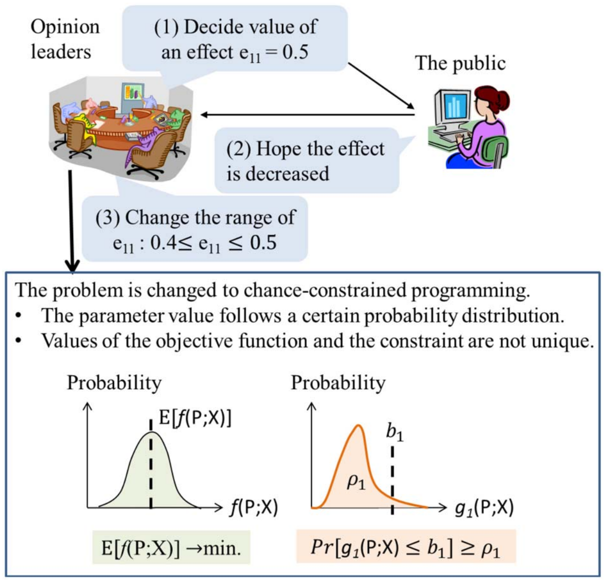

Opinion leaders
(1) Decide value of an effect e11 = 0.5
The public
(2) Hope the effect is decreased
(3) Change the range of e11 : 0.4≤ e11 ≤ 0.5
The problem is changed to chance-constrained programming.
• The parameter value follows a certain probability distribution.
• Values of the objective function and the constraint are not unique.
Probability
E[f(P;X)]
f(P;X)
Probability
b1
ρ1
g1(P;X)
E[f(P;X)] →min.
Pr[g1(P;X) ≤ b1] ≥ ρ1

Fig. 3. Formulation of chance-constrained programming in changing parameter values.

values of parameters uniquely. Opinion leaders set values of parameters based on the probability distribution, which depends on public opinion. For example, when the effect of a countermeasure for a risk is 0.5 and the public relays the opinion that the effect should be smaller, opinion leaders set values that follow a uniform distribution from 0.4 to 0.5. Generally, opinion leaders often consider both optimistic and pessimistic opinions. They set the parameter values within the range from the lowest value as the pessimistic opinion to the largest value as the optimistic opinion [14], [15]. Because opinion leaders think that all parameter values should be equally considered, they decide the probability distribution fits a uniform distribution. Even if the opinion leaders assume different uniform distributions for the same parameter, the opinion leaders use their own distributions for parameters without combining the distributions to reflect their opinions to the solutions of the programming problem. Because the programming problem has these random variables ξ, these problems are reformulated by the following chance-constrained programming:

$$\begin{aligned}\text{Minimize}\quad & \mathbb{E}[f(P(\xi);\,X)]\\ \text{s.t.}\quad & Pr\left[g_j(P(\xi);\,X)\le b_j\right]\ge\rho_j\quad (j=1,\dots m)\end{aligned}$$

where $P(\xi)$ is the parameter vector which has random variables, $E[\,\cdot\,]$ is the expectation, $Pr[\,\cdot\,]$ is the probability of satisfying the constraint and ρ_j is the lower limit of the probability. Opinion leaders solve the chance-constrained programming problems by changing the probability contribution set on each parameter, and obtain the candidates for agreement solutions.

In this paper, the purpose of this research is to solve chance-constrained programming problems in several tens of seconds to use the solutions of the problem as candidate combinations of countermeasures without undue interruption of the discussion.

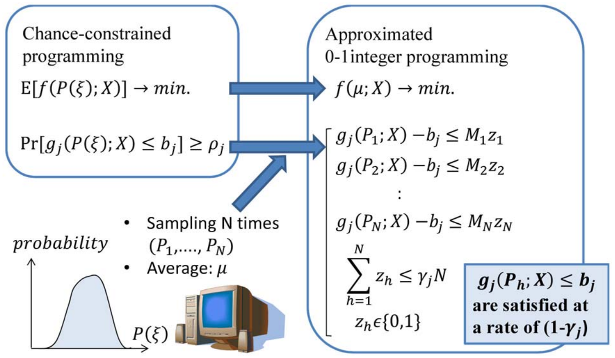

Chance-constrained programming
E[f(P(ξ); X)] → min.
Pr[g_j(P(ξ); X) ≤ b_j] ≥ ρ_j
Approximated 0-1integer programming
f(μ; X) → min.
g_j(P_1; X) − b_j ≤ M_1 z_1
g_j(P_2; X) − b_j ≤ M_2 z_2
:
g_j(P_N; X) − b_j ≤ M_N z_N
Σ_{h=1}^{N} z_h ≤ γ_j N
z_h ∈ {0,1}
g_j(P_h; X) ≤ b_j are satisfied at a rate of (1-γ_j)
• Sampling N times (P_1,...., P_N)
• Average: μ
probability
P(ξ)

Fig. 4. Sample average approximation.

C. Problems With Existing Solution Methods

There have been many studies on solving chance-constrained programming problems efficiently. When all random variables follow the independent identical distribution and all the functions are defined as linear functions, this problem can be approximated to a linear programming problem using an average and a standard deviation of the probability distribution [16]. In addition, when it is known that constraints are convex, the approximating method by convex linear constraints has been also proposed [17], [18]. However, in this research, probability distributions that the parameters follow are different, and both an objective function and the constraints are nonconvex because of the occurrence probability $A_r(e_{ir};X)$ decided by fault tree analysis.

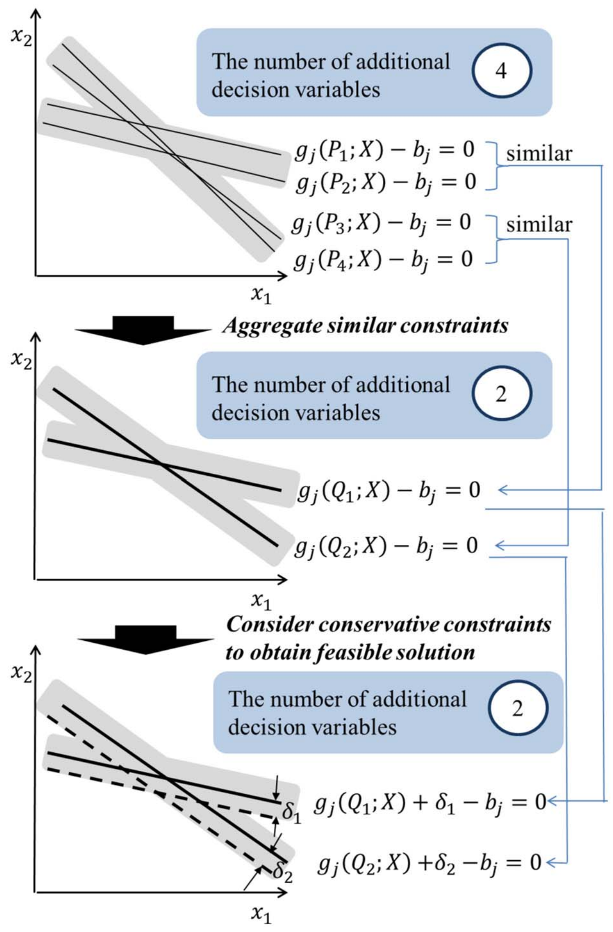
For the problems that are difficult to approximate, the sampling average approximation method has been proposed [19], [20]. Fig. 4 shows the outline of the sampling average approximation method. This method obtains values of parameters by sampling from probability distributions several times. Because the objective function has only product-sum operations of parameters, it is possible to derive expectations of the objective function by replacing the random parameters as the average of the parameters. The chance constraints are changed to constraints without random variables by using sampled values of the parameters $P_h(1\le h\le N)$. Let M_h denote a larger value than $g_j(P_h;X)\le b_j$, z_h denote an additional 0–1 decision variable, and γ_j denote $(1-\rho_j)$. The constraint is satisfied at $z_h=0$ but not satisfied at $z_h=1$. Because the sum of z_h has to be $N(1-\gamma_j)$ or smaller, the constraint $g_j(P;X)\le b_j$ is satisfied by more than a certain rate $\rho_j(=1-\gamma_j)$.

In the sampling average approximation, the more the sampling size N increases, the more the accuracy of satisfying the constraints improves. However, the additional decision variables z_h also increase. The size of the problem space becomes larger. Due to the large problem space, the computational time is still too long for the stakeholders to use the solution of the problem in their discussion.

III. High-Speed Chance-Constrained Programming Method by Constraints Aggregation

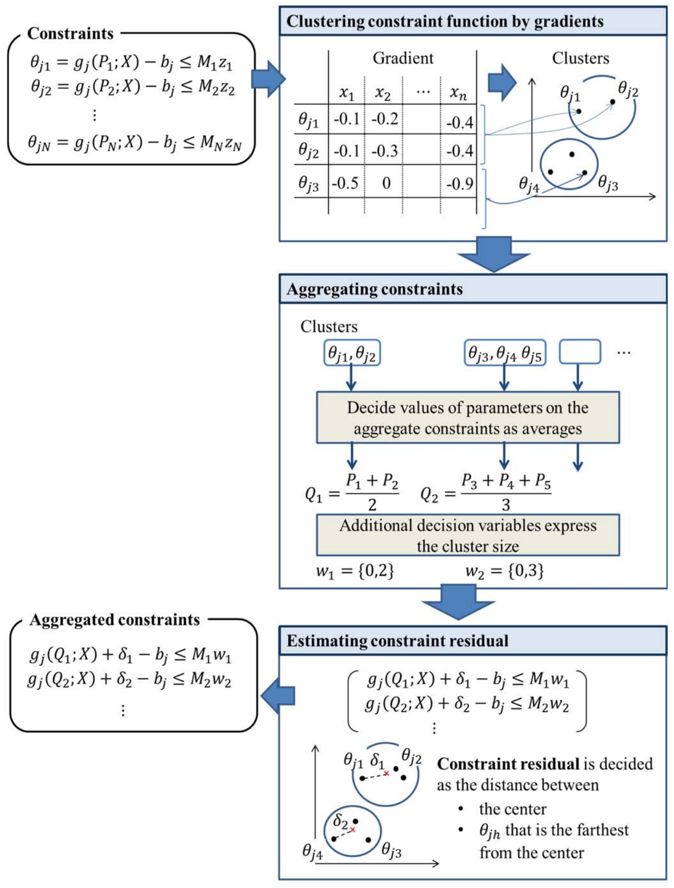
A. Approach

As described in Section II-C, the increase of the decision variable z_h makes the computational time longer in order



The number of additional decision variables 4
$g_j(P_1;X)-b_j=0$
$g_j(P_2;X)-b_j=0$
similar
$g_j(P_3;X)-b_j=0$
$g_j(P_4;X)-b_j=0$
similar
Aggregate similar constraints
The number of additional decision variables 2
$g_j(Q_1;X)-b_j=0$
$g_j(Q_2;X)-b_j=0$
Consider conservative constraints to obtain feasible solution
The number of additional decision variables 2
$g_j(Q_1;X)+\delta_1-b_j=0$
$g_j(Q_2;X)+\delta_2-b_j=0$
δ_1
δ_2
x_1
x_2

Fig. 5. Approach to aggregate the similar constraints.



Constraints
$\theta_{j1}=g_j(P_1;X)-b_j\le M_1z_1$
$\theta_{j2}=g_j(P_2;X)-b_j\le M_2z_2$
$\vdots$
$\theta_{jN}=g_j(P_N;X)-b_j\le M_Nz_N$
Clustering constraint function by gradients
Gradient
x_1 x_2 $\cdots$ x_n
θ_{j1} -0.1 -0.2 -0.4
θ_{j2} -0.1 -0.3 -0.4
θ_{j3} -0.5 0 -0.9
Clusters
θ_{j1} θ_{j2} θ_{j4} θ_{j3}
Aggregating constraints
Clusters
θ_{j1},θ_{j2}
$\theta_{j3},\theta_{j4}\,\theta_{j5}$
…
Decide values of parameters on the aggregate constraints as averages
$Q_1=\frac{P_1+P_2}{2}$ $Q_2=\frac{P_3+P_4+P_5}{3}$
Additional decision variables express the cluster size
$w_1=\{0,2\}$ $w_2=\{0,3\}$
Aggregated constraints
$g_j(Q_1;X)+\delta_1-b_j\le M_1w_1$
$g_j(Q_2;X)+\delta_2-b_j\le M_2w_2$
$\vdots$
Estimating constraint residual
$g_j(Q_1;X)+\delta_1-b_j\le M_1w_1$
$g_j(Q_2;X)+\delta_2-b_j\le M_2w_2$
$\vdots$
θ_{j1} δ_1 θ_{j2} δ_2 θ_{j4} θ_{j3}
Constraint residual is decided as the distance between
• the center
• θ_{jh} that is the farthest from the center

Fig. 6. Process of constraints aggregation.

to obtain a solution that satisfies the chance constraints. If it is possible to decrease the number of the approximated constraints with z_h, the computational time can be decreased.

Focusing on the approximated constraints, some of approximated constraints are similar because the basis of the function is the same among the approximated constraints. Fig. 5 shows our approach to aggregate the similar constraints for decreasing the number of additional decision variables.

An example of Fig. 5 shows the constraints that consist of two decision variables: 1) x_1 and 2) x_2. In the graph at the upper side of Fig. 5, there are four functions that are constraints generated by the sample average approximation. Due to the same basis of the functions, some of the constraints are also similar. By aggregating the similar constraints, the number of constraints is decreased to 2 as shown in the lower side of Fig. 5. Therefore, it is expected that aggregating the constraints decreases the computational time. Furthermore, because the aggregated constraints represent the original constraints, the proposed method can derive as a good solution as the sample average approximation.

In order to solve the chance-constrained programming problem, it is necessary to decide which constraints are replaced as aggregated constraints and to decide the parameter Q_s on the sth aggregated constraint. An average constraint is typically used instead of the constraints to be aggregated [21] but incorrectly causes infeasible solutions. The authors not only use the average of the constraint $g_j(Q_s;X)$ but also consider conservative cases where the constraint $g_j(Q_s;X)$ is increased to $g_j(Q_s;X)+\delta_s$. The authors call δ_s "constraint residual."

The following sections describe the method of constraints aggregation with estimating constraint residual.

B. Constraints Aggregation With Estimating Constraint Residual

In order to aggregate the constraints, the proposed method needs to choose constraints to be aggregated and to decide parameter values in the aggregated constraints. The process of the constraints aggregation is shown in Fig. 6.

The process of the constraints aggregation consists of the following functions.

1) Clustering Constraint Functions by Gradients: As shown in Fig. 5, the similarity of the constraints is equivalent to the similarity of the gradient of the constraint function. So, the proposed method calculates the gradient of the decision variable x_i in the constraint function $\theta_{jh}=g_j(P_h;X)-b_j$. Because the value of the decision variable is either 0 or 1, the gradient grad_{jih} of x_i in θ_{jh} is decided by the following:

$$\text{grad}_{jih}=\left.\frac{\partial\theta_{jh}}{\partial x_i}\right|_{X=B_i}-\left.\frac{\partial\theta_{jh}}{\partial x_i}\right|_{X=0}.$$

B_i is a vector where x_i is 1 and the others are 0. By calculating grad_{jih}, it is possible to obtain the vectors of grad_{jih} on θ_{jh} as shown in Fig. 6.

Based on the vectors of grad_{jih}, the proposed method makes clusters of the constraint functions. The proposed method applies k-means clustering as a typical clustering method [22]. Before clustering, we have to decide the number of clusters. In general, in order to determine the number of clusters, Bayesian information criterion and Akaike information criterion are widely used [23], [24]. k-means clustering assigns θ_{jh} to the clusters randomly. Calculating the distance between θ_{jh} and the center of the cluster, k-means clustering reassigns θ_{jh} to the cluster where the distance between the θ_{jh} and the center is the smallest. The authors decide the center of the cluster as the average values of grad_{jih} that belong to the cluster. Let $\text{Cen} = \{\text{Cen}_1, \cdots \text{Cen}_n\}$ denote the vector of the average values of the gradients

$$\text{Cen}_i = \frac{\text{Sum of } \text{grad}_{jih} \text{ in the cluster}}{\text{The number of } \theta_{jh} \text{ in the cluster}}.$$

The authors define the distance between θ_{jh} and Cen as the Euclidean distance

$$\text{distance}(\theta_{jh}, \text{Cen}) = \sqrt{\left(\text{grad}_{j1h} - \text{Cen}_1\right)^2 + \cdots + \left(\text{grad}_{jnh} - \text{Cen}_n\right)^2}.$$

When no θ_{jh} are reassigned to the clusters (i.e., the closest cluster for θ_{jh} is not changed by any reassignment), the proposed method determines the clusters of θ_{jh} for the aggregation.

2) Aggregating Constraint: The proposed method replaces the constraints with the aggregated constraint that consists of the average values of the parameters in the constraints in the cluster. Let Q_s denote the parameter vector of the aggregated constraint in the sth cluster Cl_s that is a set of h in the cluster. Q_s is calculated by the following formula:

$$Q_s = \frac{\sum_{h \in \text{Cl}_s} P_h}{|\text{Cl}_s|}$$

where $|\text{Cl}_s|$ indicates the number of the constraints in the sth cluster.

Next, the additional decision variable z_h has to be changed because satisfying the aggregated constraint is equivalent to satisfying multiple constraints in the cluster. So, the additional decision variable w_s that considers the number of constraints in the cluster is as follows:

$$w_s \in \{0, |\text{Cl}_s|\}.$$

3) Estimating the Constraint Residual: Through the above steps, $g_j(Q_s; X) - b_j$ and $M_s w_s$ can be decided, but δ_s has not been decided yet. When δ_s is large, $g_j(Q_s; X) - b_j$ has to be smaller, which enables satisfaction of the chance constraints at a high possibility. The difference among aggregated constraints at step 2 depends on the distance in the cluster. So, δ_s is decided as the largest distance$(\theta_{jh}, \text{Cen})$ in the hth cluster. Finally, the aggregated constraint is described as follows:

$$g_j\left(Q_s;\ X\right) + \max_h \text{distance}(\theta_{jh}, \text{Cen}) - b_j \le M_s w_s.$$

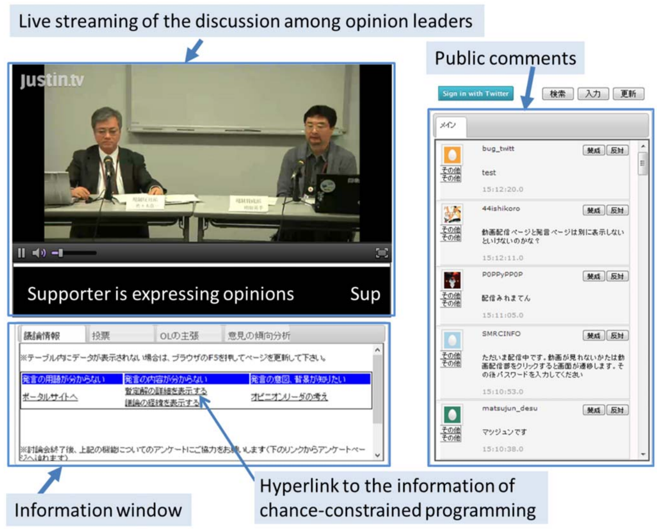

Live streaming of the discussion among opinion leaders
Public comments
Supporter is expressing opinions
Information window
Hyperlink to the information of chance-constrained programming

Fig. 7. Screen to display discussion and input opinions (two opinion leaders are discussing).

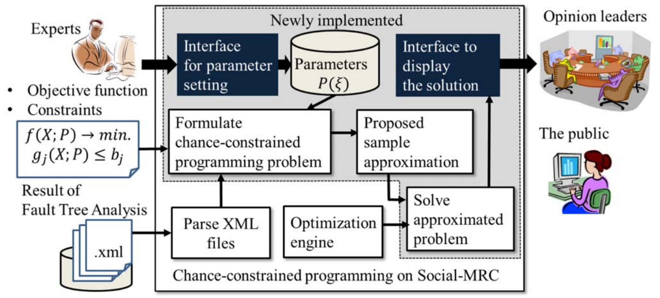

Experts
Objective function
Constraints
$f(X;P) \to min.$
$g_j(X;P) \le b_j$
Result of Fault Tree Analysis
.xml
Newly implemented
Interface for parameter setting
Parameters $P(\xi)$
Interface to display the solution
Formulate chance-constrained programming problem
Proposed sample approximation
Solve approximated problem
Parse XML files
Optimization engine
Chance-constrained programming on Social-MRC
Opinion leaders
The public

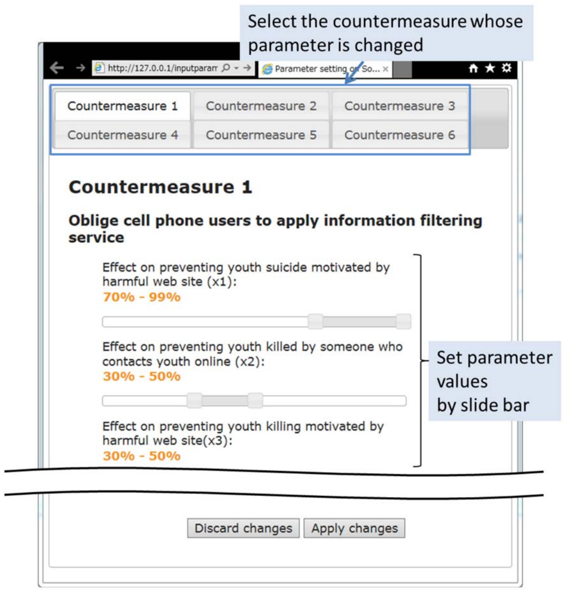
Fig. 8. Block diagram of the implemented system.

IV. System Implementation

We have implemented the chance-constrained programming system on social-MRC. During the discussion to reach consensus, the opinion leaders and the public check the screen of the social-MRC shown in Fig. 7. All the descriptions on the screens are written in Japanese, but some of them are translated to English. When the opinion leaders find the public request for changing parameter values on the screen, the opinion leaders change the parameter values and solve the chance-constrained programming problem with the changed parameter values.

Fig. 8 shows the block diagram of the implemented system to change the parameter values and to solve the chance-constrained programming problem. Based on the opinions from the public shown in the screen, opinion leaders set parameter values to an objective function and constraints via interface for parameter setting. Fig. 9 shows the screen for inputting parameter values. After the opinion leaders decide which parameter values are changed, the opinion leaders clicks one of tabs that indicate the countermeasures of the parameter values on the top of the screen. The opinion leader sets the range of the parameter value by slide bar.

By applying the change of the parameter values, the problem that the stakeholders discuss is changed to a chance-constrained programming problem. The proposed approximation method is applied to the chance-constrained


Select the countermeasure whose parameter is changed
http://127.0.0.1/inputparam
Parameter setting of So...
Set parameter values by slide bar
Discard changes
Apply changes

Countermeasure 1 | Countermeasure 2 | Countermeasure 3
Countermeasure 4 | Countermeasure 5 | Countermeasure 6

Countermeasure 1

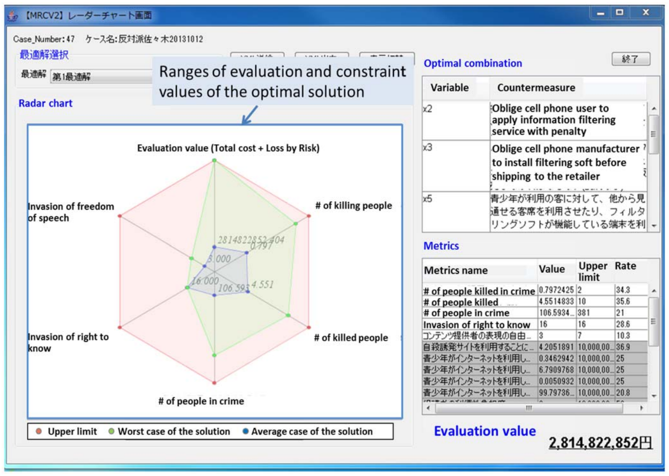
Oblige cell phone users to apply information filtering service

Effect on preventing youth suicide motivated by harmful web site (x1):
70% - 99%

Effect on preventing youth killed by someone who contacts youth online (x2):
30% - 50%

Effect on preventing youth killing motivated by harmful web site(x3):
30% - 50%

Fig. 9. Screen for inputting parameter values.


Ranges of evaluation and constraint values of the optimal solution
Radar chart
Evaluation value (Total cost + Loss by Risk)
# of killing people
# of killed people
# of people in crime
Invasion of right to know
Invasion of freedom of speech
2814822852.404
0.797
3.000
16.000
106.593
4.551
Upper limit
Worst case of the solution
Average case of the solution
終了

【MRCV2】レーダーチャート画面

Case_Number:47 ケース名:反対派佐々木20131012

最適解選択

最適解 第1最適解

Optimal combination

Variable	Countermeasure
x2	Oblige cell phone user to apply information filtering service with penalty
x3	Oblige cell phone manufacturer to install filtering soft before shipping to the retailer
x5	青少年が利用の客に対して、他から見通せる客席を利用させたり、フィルタリングソフトが機能している端末を利

Metrics

Metrics name	Value	Upper limit	Rate
# of people killed in crime	0.7972425	2	34.3
# of people killed	4.5514833	10	35.6
# of people in crime	106.5934...	381	21
Invasion of right to know	16	16	28.6
コンテンツ提供者の表現の自由...	3	7	10.3
自殺誘発サイトを利用することに...	4.2051891	10,000,00...	36.9
青少年がインターネットを利用し...	0.3462942	10,000,00...	25
青少年がインターネットを利用し...	6.7909768	10,000,00...	25
青少年がインターネットを利用し...	0.0050932	10,000,00...	25
青少年がインターネットを利用し...	99.79736...	10,000,00...	20.8

Evaluation value 2,814,822,852円

Fig. 10. Screen for showing information about the solution.

programming, and the approximated problem is solved by the existing optimization engine. Finally, values of the objective function and the constraint values are shown to the stakeholders and the public via the interface to display the solution.

Fig. 10 shows the screen for displaying information about the solution. The solution is shown at the top right; countermeasures 2, 3, 5, and so on are selected as a solution. The evaluation and constraints values are shown by a radar chart at the left side. The axes indicate the values of the evaluation and constraints on the original programming problem before being transformed to the chance-constrained programming problem. It is necessary to consider that the evaluation value and the constraint values are not unique because of changing parameter values. On the radar chart, it is possible to find two kinds of graphs to indicate the evaluation value and the constraint values on the solution of the chance-constrained programming problem. The innermost graph indicates average values of the evaluation and the constraints and the second innermost graph indicates the worst values of the evaluation and the constraints. The detailed values are shown in the bottom right table.

V. Evaluation Experiment

A. Target of Experiment

The targets of the experiment are to reach consensus on countermeasures for: 1) an information leakage in an information service industry [25] and 2) a spread of harmful information to youth in a wireless carrier industry [26]. The authors call them "problem 1" and "problem 2," respectively.

1) *Information Leakage in an Information Service Industry:* Many incidents of personal information leakage happen and damage the companies that provide services based on the personal information. The countermeasures for the information leakage are installing firewalls, introducing surveillance cameras to the server room, prohibiting e-mail file attachments without authorization, and so on. Some of the countermeasures invade the privacy or are inconvenient for workers in the company. Opinion leaders of the company and the workers need to discuss to reach a consensus. The number of the countermeasure is 15 and the number of risks is 14. This problem consists of the same formulas as $f(P;X)$, $g_1(P;X)$, $g_2(P;X)$ in (1)–(3) in addition to the following formulas:

$$g_3(P;X) = \sum_{i=1}^{15} \mathrm{inc}_i x_i \le b_3 \tag{4}$$

$$g_4(P;X) = \sum_{i=1}^{15} \mathrm{pri}_i x_i \le b_4 \tag{5}$$

where $g_3(P;X)$ indicates the constraint that the total of inconvenience inc_i by the ith countermeasure has to be b_3 or less, and $g_4(P;X)$ indicates the constraint that the total of the privacy invasion pri_i by the ith countermeasure has to be b_4 or less.

2) *Spread of Harmful Information to Youth in a Wireless Carrier Industry:* Using a cell phone, many youth that do not have sufficient computer literacy can get information via Internet. This enables the youth to access harmful information, such as Internet suicide, child prostitution, and so on. In order to protect youth from such harmful information, the Japanese government and wireless carrier industries discuss which kinds of content filtering are necessary. The number of countermeasures on the content filtering is 15 and the number of risks, such as Internet suicide, is 6. This problem consists of the same formula as $f(P;X)$ in addition to the following formulae:

$$g_5(P;X) = L_3A_3(e_{i3};X) \le b_5 \tag{6}$$

$$g_6(P;X) = L_1A_1(e_{i1};X) + L_2A_2(e_{i2};X) \le b_6 \tag{7}$$

$$\begin{aligned} g_7(P;X) = {} & L_4A_4(e_{i4};X) + L_5A_5(e_{i5};X) \\ & + L_6A_6(e_{i6};X) \le b_7 \end{aligned} \tag{8}$$

$$g_8(P;X) = \sum_{i=1}^{15} \mathrm{right}_i x_i \le b_8 \tag{9}$$

$$g_9(P;X) = \sum_{i=1}^{15} \mathrm{free}_i x_i \le b_9 \tag{10}$$

TABLE I
COMPUTATIONAL TIME AND EVALUATION VALUE BY THE BRUTE FORCE METHOD

	Problem 1	Problem 2
Computational time [minutes]	914	442
Evaluation value	2.241×10^8	4.80×10^8

where $g_5(P;X)$, $g_6(P;X)$, and $g_7(P;X)$ indicate the constraints on the number of youth's killing people, the number of killed youth, and the number of youth involved in crimes, respectively. $g_8(P;X)$ indicates the constraint that the total of right-to-know invasion right$_i$ by the ith countermeasure must be b_8 or less, and $g_9(P;X)$ indicates the constraint that the total invasion of freedom of speech free$_i$ by the ith countermeasure must be b_9 or less.

The parameter values are estimated by experts and shown in [25] and [26]. According to discussions among experts and opinion leaders, the authors found that it is difficult to determine a unique value for the effects e_{ir} of each countermeasure. Therefore, experts set the range of the effect e_{ir}; the maximum value of e_{ir} is the same as the initial value set by the experts and the minimum value of e_{ir} is 90% of the initial value. The authors assume that e_{ir} follows a uniform distribution within the range $[0.9 \times e_{ir}, e_{ir}]$. Therefore, the problems must be solved as a chance-constrained programming problem, including chance constraints of $Pr[g_1(P(\xi);X) \le b_1] \ge \rho_1, Pr[g_5(P(\xi);X) \le b_5] \ge \rho_5, Pr[g_6(P(\xi);X) \le b_6] \ge \rho_6, Pr[g_7(P(\xi);X) \le b_7] \ge \rho_7$. The lower bound of the probability is set to $\rho_1, \rho_5, \rho_6, \rho_7 = 0.95$.

Based on these conditions, the authors implement our proposed method on a computer that has a Windows 2008 Server operating system, CPU Intel Xeon CPU E5-2450 2.10 GHz $\times$ 2, and 128 GB memory. Using two kinds of the above described problems, the authors evaluate the effectiveness of the proposed method by comparing to the sample average approximation method [19]. In applying both methods, the number of sampling is set to $N = \{10, 20, 30, \ldots, 100\}$. The number of clustering in the proposed method is set to 5 or 10. The authors use Mathematica [27] to solve the 0–1 programming problem that is approximated by either the sampling average approximation or the proposed method, and Weka [28] to apply k-means for clustering constraints.

The approximation method can not always derive the optimal solution to minimize the objective function under constraints. In order to evaluate the solutions by the proposed method, the authors apply the brute force method to find the optimal solution from all the candidate solutions ($= 2^{15}$ solutions) while judging whether each candidate solution satisfies the chance constraint by sampling the parameter of the effects 100 times. Table I shows the computational time and evaluation value that are obtained by the brute force method.

As shown in Table I, the computational time by the brute force method is so long that it is impossible to use during the discussion. Because a small computational time and good evaluation value with fewer infeasible solutions are expected from the proposed method, the authors set the evaluation criteria as follows.

1) Computational time.
2) *Error Rate on the Evaluation:* The authors have ten trials of the approximation methods, the sample average approximation and the proposed method for each number of sampling times and each cluster size to get the average of the evaluation value $\mathbb{E}[f(P(\xi);X)]$. The error rate as the difference between the evaluation value by an approximation method and the brute force method is calculated by the following formula:

$$\text{Error rate} = \frac{\mathbb{E}[f(P(\xi);X)] \text{ by an approximation method}}{\mathbb{E}[f(P(\xi);X)] \text{ by the brute force method}}.$$

3) *Rate of Feasible Solutions:* In order to evaluate how often the proposed method outputs feasible solutions, the rate of feasible solutions for ten trials of the approximation methods is calculated by the following formula:

$$\text{Rate of feasible solutions} = \frac{\text{The number of feasible solutions}}{\text{The number of trials}(=10)}.$$

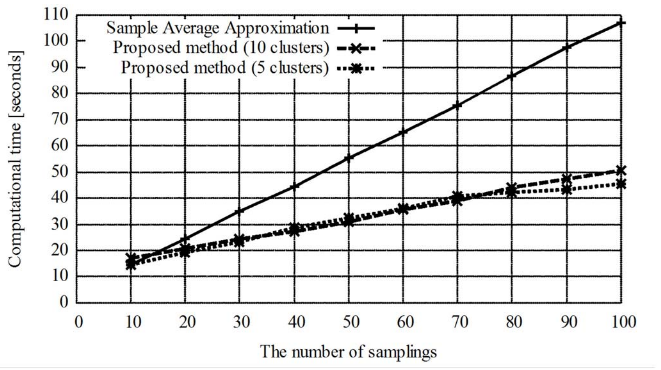


Fig. 11. Computational time to solve problem 1.

B. Experimental Result

First, the authors discuss the computational time in order to judge whether the proposed method is useful enough to apply for consensus support. Figs. 11 and 12 show the computational time to solve the problems 1 and 2 by the method with the sample average approximation and the proposed method.

According to Figs. 11 and 12, the computational time by the methods are increased linearly depending on the number of sampling. The computational time by the method with sample average approximation is increased more than by the proposed method. When the number of sampling is over 60, the sample average approximation takes 1 min. By comparing computational time using the sample average approximation in Figs. 11 and 12, it is confirmed that the computational time to solve problem 2 is longer than to solve problem 1. This is because the problem 2 includes three chance constraints.

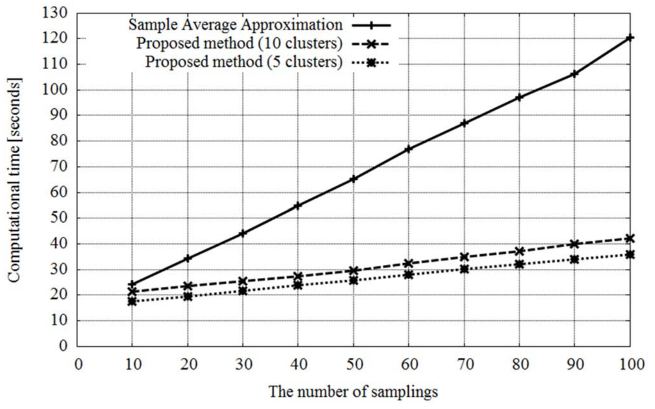


Fig. 12. Computational time to solve problem 2.

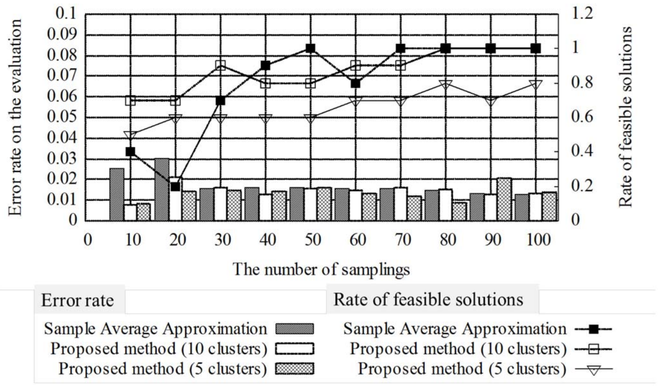


Fig. 13. Error rate on the evaluation and rate of feasible solutions in the case of problem 1.

The more chance constraints there are, the more approximated constraints are derived by the sample average approximation. This makes the computational time longer without aggregation of derived constraints. On the other hand, the proposed method can keep the computational time under 1 min when the number of sampling is below 100, which enables support reaching of the consensus. Small numbers of clusters decrease the computational time, but the amount of the decrease is not large.

Next the authors discuss the error rate on the evaluation and the rate of feasible solutions. Figs. 13 and 14 show the error rate on the evaluation and the rate of feasible solutions in the case of problems 1 and 2, respectively.

1) *Error Rate:* Error rates are shown in the bar chart. When the number of sampling is increased, the error rates are often decreased. This is because only tight constraints may be derived by a small number of sampling. Large number of sampling derives various kinds of constraints, including tight constraints and loose constraints, which can decreases the error rate.

   Through comparing the error rates by the proposed method using five clusters and ten clusters, the error rate by the proposed method using three clusters is larger than the proposed method using ten clusters. When the number of the clusters is set to a small value, different constraints are aggregated to the same cluster, which makes the error rate larger.

   According to the error rates by the sample average approximation and by the proposed method using ten clusters, the error rate by the proposed method is a little bit larger than the error rate by the sample average approximation when the number of sampling is large. The reason why the error rate is increased is that constraint residual in the proposed method considers pessimistic situations. Constraint residual for pessimistic situations increases the error rate and the rate of feasible solutions.

2) *Rate of Feasible Solutions:* The rates of feasible solutions are shown in line charts. The larger number of sampling tends to increase the rate of feasible solutions because various constrains are derived by sampling for chance constraints for obtaining the feasible solutions.

   As is the case in error rates, the small number of clusters decreases the rate of feasible solutions due to different constraints in the same cluster. The proposed method using ten clusters increases the rate of feasible solutions to 1.0 in both cases, which is better than the sample average approximation.

   By comparing the rate of feasible solutions in the cases of problems 1 and 2, it is confirmed that the rate in the cases of problem 2 is worse than one in the cases of problem 1. Because problem 2 has three chance constraints, the sample average approximation has to deal with the constraints that are triple the number of sampling. Due to many constraints, it is difficult to calculate the feasible solution. So, the rate of the feasible solution is small in using the sample average approximation.

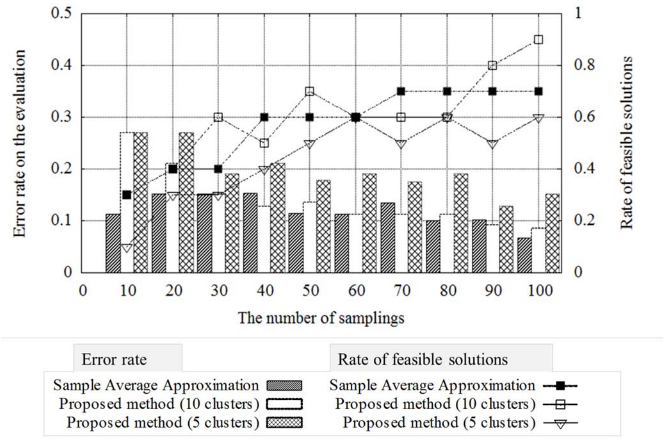


Fig. 14. Error rate on the evaluation and rate of feasible solutions in the case of problem 2.

Through the above comparative analysis on the experimental results, it is confirmed that the proposed method can decrease the computational time and keep both the error rate and the rate of the feasible solutions. The chance-constrained programming problem can be solved several times in one discussion with changing parameter values or limit values of the constraints. Because waiting time for obtaining solutions can be decreased by the proposed method, the stakeholders can use more time for improving the discussion.

VI. Conclusion

The authors have developed the social-MRC to support reaching the social consensus on IT risk countermeasures. The social-MRC solves the 0–1 integer programming problem of the risk countermeasures, and displays the solutions to the stakeholders. However, it is difficult for the stakeholders to determine a unique value for each parameter on the 0–1 integer programming problem. The authors reformulated the problem as the chance-constrained programming problem. Because solving the chance-constrained programming problem takes much computational time using the sample average approximation, the computational time must be decreased for the use in the stakeholders' discussion.

In this paper, the authors proposed the chance-constrained programming method by constraints aggregation to decrease the computational time. Some of the constraints that are generated by the sample average approximation are similar to each other. The similarity between the constraints are decided by gradients of the decision variables. By clustering the constraints based on the vectors that consist of the gradients, the average value of each parameter on the constraints in a cluster is set to the value of the parameter on the aggregated constraint. In addition, the authors introduced the constraint residual on the aggregated constraints in order to get the feasible solutions.

Experimental results on two actual cases of information leakage and spread of harmful information showed that the proposed method decreased the computational time to 1 min while obtaining as good evaluation values as the sample average approximation. On the other hand, the proposed method improves the rate of the feasible solutions. Therefore, the discussion among stakeholders becomes more efficient by the proposed method than ever before.

References

[1] T. R. Peltier, *Information Security Risk Analysis*. Boca Raton, FL, USA: Auerbach, 2011.

[2] W. H. Baker and L. Wallace, "Is information security under control?: Investigating quality in information security management," *IEEE Secur. Privacy*, vol. 5, no. 1, pp. 36–44, Jan./Feb. 2007.

[3] ISACA, *Risk IT*. Rolling Meadows, IL, USA: Inf. Syst. Audit Control Assoc., 2009.

[4] R. Sasaki *et al.*, "Social consensus formation support system concerning IT risk countermeasures," *Int. J. Inf. Process. Manage.*, vol. 2, no. 2, pp. 2562–2574, 2011.

[5] L. A. Wolsey, *Integer Programming*. New York, NY, USA: Wiley, 1998.

[6] D. Vose, *Risk Analysis: A Quantitative Guide*. New York, NY, USA: Wiley, 2000.

[7] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. New York, NY, USA: Springer, 2011.

[8] X. Li, *Credibilistic Programming: An Introduction to Models and Applications*. New York, NY, USA: Springer, 2013.

[9] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Manage. Sci.*, vol. 6, no. 1, pp. 73–79, 1959.

[10] D. A. Belsley and E. Kontoghiorghes, *Handbook of Computational Econometrics*. Chichester, U.K.: Wiley, 2009.

[11] J. Cardinal, S. Fiorini, and G. Joret, "Minimum entropy combinatorial optimization problems," *Theory Comput. Syst.*, vol. 51, no. 1, pp. 4–21, 2012.

[12] P. J. Brooke and R. F. Paige, "Fault trees for security system design and analysis," *Comput. Security*, vol. 22, no. 3, pp. 256–264, 2003.

[13] M. Stamp, *Information Security: Principles and Practice*. Hoboken, NJ, USA: Wiley, 2005.

[14] M. Amer, T. U. Daim, and A. Jetter, "A review of scenario planning," *Futures*, vol. 46, pp. 23–40, Feb. 2013.

[15] J. Mun, *Modeling Risk: Applying Monte Carlo Risk Simulation, Strategic Real Options, Stochastic Forecasting, and Portfolio Optimization*. Hoboken, NJ, USA: Wiley, 2010.

[16] D. L. Olson and S. R. Swenseth, "A linear approximation for chance-constrained programming," *J. Oper. Res. Soc.*, vol. 38, no. 3, pp. 261–267, 1987.

[17] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM J. Optim.*, vol. 17, no. 4, pp. 969–996, 2006.

[18] F. Oldewurtel, C. Jones, and M. Morari, "A tractable approximation of chance constrained stochastic MPC based on affine disturbance feedback," in *Proc. 47th IEEE Conf. Decis. Control*, Cancún, Mexico, 2008, pp. 4731–4736.

[19] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro, "Sample average approximation method for chance constrained programming: Theory and applications," *J. Optim. Theory Appl.*, vol. 142, no. 2, pp. 399–416, 2009.

[20] M. Samejima and R. Sasaki, "Approximation method for chance-constrained programming of social consensus formation concerning IT risk countermeasure," in *Proc. 2012 IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Seoul, Korea, pp. 2751–2755.

[21] K. Sato, M. Samejima, and R. Sasaki, "Chance-constrained programming method by constraints aggregation for social consensus making on IT risk countermeasure," in *Proc. 2013 IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Manchester, U.K., pp. 4789–4793.

[22] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. London, U.K.: Elsevier, 2005.

[23] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Mach. Learn.*, Stanford, CA, USA, 2000, pp. 727–734.

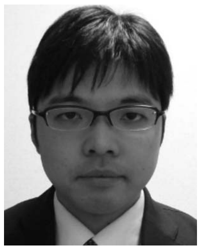
[24] C. A. Sugar and G. M. James, "Finding the number of clusters in a data set: An information theoretic approach," *J. Amer. Stat. Assoc.*, vol. 98, no. 463, pp. 750–763, 2003.

[25] M. Taniyama *et al.*, "Application of 'multiple risk communicator' to the personal information leakage problem," in *Proc. World Acad. Sci. Eng. Technol.*, vol. 45, 2006, pp. 284–289.

[26] M. Ohkawara *et al.*, "Application of social consensus support system for IT risk measure 'social-MRC' to the information filtering issue for children," in *Proc. 3rd Int. Conf. e-Educ. e-Bus. e-Manage. e-Learn.*, 2012, pp. 158–167.

[27] (Mar. 2, 2014). *Mathematica*. [Online]. Available: http://www.wolfram.com/mathematica/

[28] (Mar. 2, 2014). *Weka*. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/



Masaki Samejima (M'09) received the master's and Ph.D. degrees in information science from Osaka University, Suita, Japan, in 2007 and 2008, respectively.

He was with Hitachi, Ltd., Tokyo, Japan, from 2007 to 2009. He is currently an Assistant Professor with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University. His current research interests include risk-based design and management of information systems.

Ryoichi Sasaki (M'82–LM'15) received the B.S. degree in health science and the Ph.D. degree in system engineering from the University of Tokyo, Bunkyō, Japan, in 1971 and 1981, respectively.

From 1971 to 2001, he was engaged in the research and research management on systems safety, network management, and information security at Systems Development Laboratory, Hitachi, Ltd., Tokyo, Japan. Since 2001, he has been a Professor of School of Engineering, Tokyo Denki University, Tokyo.