ITC 4/47

Journal of Information Technology and Control
Vol. 47 / No. 4 / 2018
pp.728-738
DOI 10.5755/j01.itc.47.4.20103

Modified L-Shaped Decomposition Method with Scenario Aggregation for a Two-Stage Stochastic Programming Problem

Received 2018/02/05 | Accepted after revision 2018/10/31

http://dx.doi.org/10.5755/j01.itc.47.4.20103

Modified L-Shaped Decomposition Method with Scenario Aggregation for a Two-Stage Stochastic Programming Problem

Ana Ušpurienė

Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223 Vilnius, Lithuania, e-mail: a.uspuriene@gmail.com

Leonidas Sakalauskas, Gediminas Gricius

Klaipeda University, H. Manto str. 84, LT- 92294 Klaipeda, Lithuania, e-mails: leonidas.sakalauskas@mii.vu.lt, gediminas.gricius@ku.lt

Corresponding author: a.uspuriene@gmail.com

This paper introduces a method which is developed to solve two-stage stochastic programming problems in which first-stage region is unbounded and cannot be solved using traditional decomposition. We are using our proposed L-shaped decomposition method modification to solve such problems. In order to achieve a more accurate result, the number of scenarios generated in the optimization process must be large enough. If there is a large number of target variables, optimization takes a long time and uses a lot of resources. Thus, in order to reduce the number of iterations of the optimization process, the amount of resources used, and the calculating time needed to get the optimal solution, the aggregation approach is applied. This paper also presents results of our research on optimal parameters setting of the proposed method.

KEYWORDS: Stochastic programming, scenario analysis, aggregation, optimization under uncertainty, decomposition method, IBM ILOG CPLEX solver.

1. Introduction

Most systems that need to be controlled or analyzed involve some level of uncertainty [11] that can be included in the model of the task in many ways. One of them is to model uncertain quantities of the model as random variables. When we solve decision-making problems, we can use stochastic optimization models. In recent years, interest in stochastic optimization models has increased. For example, Schulzea et al. [13] proposed such a model and new reformulation of classical algorithms for hydro-thermal unit commitment. A solution of such two-stage or multi-stage stochastic mixed-integer optimization problems is directly computationally intractable for large instances, therefore, alternative approaches are required [13]. Schulzea et al. [13] use Dantzig-Wolfe reformulation to decompose the stochastic problem by scenarios. Another often used algorithm is Benders decomposition, also known as L-shaped. This algorithm has been successfully applied to a wide range of difficult optimization problems [9]. Rahmaniani et al. [9] discussed in their papers about the classical algorithm, the impact of problem formulation on its convergence, and the relationship to other decomposition methods [9]. Adequate model creation is a very important part of the simulation process. Wang et al. [18] focus on the modelling methodology of two-stage stochastic programming, and present some applications of two-stage stochastic programming in chapter five of their book [18].

In this paper, we consider a two-stage stochastic linear programming (SLP) [20] task, where the variables of the second stage depend on a random parameter. By random process simulation, random scenarios are generated in accordance with the multi-normal distribution. In solving this problem, a modified L-shaped algorithm is usually applied. To achieve a more precise result, the number of scenarios generated during the optimization process must be large enough. When the number of task variables is high, optimization takes a lot of time and consumes a lot of resources, therefore, it is necessary to look for new ways of solving similar problems. For example, Fountoulakis and Gondzio [4], in their paper, presented a rigorously defined generator that allows the control of dimensions, conditioning, and sparsity of the problem when the conditioning of the problem changes and its dimensions increase up to one trillion [4]. The proposed generator has very low memory requirements and scales well with the dimensions of the problem [4]. Ogbe and Li [8] proposed a new cross decomposition method combining two classical decomposition methods [8]. Their method outperforms Benders decomposition when the number of scenarios is large [8]. In order to minimize the number of iterations in the optimization process, the amount of resources used, and the time to calculate the optimal solution, we, in our research, analysed the scenario aggregation method applied to solving the two-stage SLP problem.

For simulation and optimization, specialized software packages are often used, and the integration of these packages is very important. Vamanana et al. [17] demonstrated the mechanics of integrating two commonly used Operations Research software packages, IBM ILOG CPLEX (often informally referred to, simply, as CPLEX) and ARENA [17]. It is also important to properly select the model solution parameters. For example, Sun et al. [14] use CPLEX a branch-and-cut method in their work for optimization of firing transition sequences, based on the Mixed-timed Petri net [15]. In our realizations, we used the CPLEX optimization package (see Section 6).

The initial solution must be selected using the L-shaped decomposition algorithm. In order to obtain this solution in the classical algorithm, the first-stage task is solved. However, if the first-stage region is unbounded, then this task has no solution. Thus, if the solution of the first-stage or second-stage task does not exist, but the solution of the stochastic task exists, the classic L-shaped algorithm is not suitable, as there may not exist a feasible cut. However, the problem can be solved by the proposed modification of the classical algorithm [16].

This often occurs in business when dealing with investment objectives if the return of the first stage is negative. In this paper, we deal specifically with such tasks.

2. A Two-Stage SLP Problem

It is mostly impossible to exactly evaluate individual realization factors of solutions in planning tasks. In these cases, the solutions should be corrected during their realization. A two-stage stochastic programming model allows us to evaluate the uncertainty and get solutions that require minimal correction costs [1, 7, 12].

In formulating a two-stage SLP task, the vector for the first-stage variables is denoted by x. The first-stage variable vector is found before a concrete random parameter value is known. Later on, in view of the realization of a random vector, the second-stage solution y is adjusted [1].

A two-stage SLP problem can be formulated as follows:

$$F(x) = \min_{x \in D_1 \subset R_+^n} \left[c^T x + E\left(f(x, \xi) \right) \right] \tag{1}$$

The values of a feasible set are:

$$D_1 = \left\{ x | Ax = b, x \in R_+^{n1} \right\} \tag{2}$$

The second-stage objective function is defined as follows:

$$f(x, \xi) = \min_y \left\{ q^T y | Wy + Tx = h, y \in R_+^{n2} \right\}, \tag{3}$$

where $n1$ is the number of variables in the first-stage, $m1$ is the number of constraints in the first-stage, $n2$ is the number of variables in the second-stage, $m2$ is the number of constraints in the second-stage, $c[n1]$ is the vector of the objective function coefficients of the first-stage variables, $b[m1]$ is the vector of the right-hand side of the first-stage constraints, $q[n2]$ is the vector of the objective function coefficients of the second-stage variables, $h[m2]$ is the vector of the right-hand side of the second-stage constraints, $A[m1 \times n1]$ is the matrix of the first-stage constraint coefficients, $T[m2 \times n2]$ is the matrix of the second-stage constraint coefficients of the first-stage variables, $W[m2 \times n2]$ is the matrix of the second-stage constraint coefficients of the second-stage variables.

Assume that the feasible set D_1 is not empty. The vector h is random. Suppose that the solution of the second-stage problem (see Equation (3)) and the meaning of the function f almost certainly exist and are finite.

When a random parameter ξ is defined by a discrete distribution, where the probabilities of each scenario are denoted $p_j, j = \overline{1, K}$, where K is the number of scenarios, and the deterministic equivalence of the model can be written as follows:

$$\min_x \left[c^T x + \sum_{j=1}^{K} p_j q^T y^j \right], \tag{4}$$

where

$$\begin{aligned} & Ax = b, \\ & Tx + Wy^j = h^j, \\ & x \geq 0, \ \ y^j \geq 0, \ \ j = \overline{1, K}, \end{aligned} \tag{5}$$

x is the vector of the first-stage variables, y is the vector of the second-stage variables, which can be written as the linear programming block task:

$$\begin{cases} Ax & & & & = b, \\ Tx & +Wy^1 & & & = h^1, \\ Tx & & +Wy^2 & & = h^2, \\ \cdots & & \cdots & & \cdots \\ \cdots & & \cdots & & \cdots \\ Tx & & & +Wy^K & = h^K, \\ x \geq 0, & y^1 \geq 0, & y^2 \geq 0, & \cdots \ , & y^K \geq 0. \end{cases} \tag{6}$$

In the solution of the two-stage SLP problem described above, a modified L-shaped algorithm was used. The scenarios were generated according to the normal law $Norm(\mu_k, \sigma_k)$, where μ_k is the arithmetic mean, and σ_k is mean square deviation (standard deviation).

The values of random variables are often observed for a long time consisting of several stages. During long-term planning, decisions are made at the end of each stage. Decisions made in one stage can have a significant impact on subsequent decisions.

In most of the methods used for solving SLP tasks, a decomposition idea is applied. The initial task having large measurements (variables, constraints) is decomposed into separate smaller tasks solutions which are later integrated into the general solution of the task. The methods of Dantzig-Wolfe, Benders, the stochastic decomposition, and other methods are applied most widely. In order to apply multi-stage task decomposition algorithms to solve two-stage tasks, it is necessary to perform a certain adaptation of algorithms.

A modified decomposition algorithm of a two-stage SLP problem is described below, applied in the case where solution of the first-stage task does not exist.

3. A Modified Decomposition Algorithm of a Two-Stage SLP Problem

The algorithm is iterative and consists of several steps. Its iterations are repeated until the required accuracy is obtained. The algorithm is based on the Sample Average Approximation [14] and decomposition method described by Bierge et al. [1].

Before the first iteration the initial values are defined: $r = s = \nu = 0$, where r is the number of constraints for feasible cut, s is the number of constraints for optimality cut, ν is the iteration number. All the algorithm steps are described below.

The start of the algorithm.

Step 1.

Before the first iteration, the initial point task is solved:

$$\min_x \left\{ c^T x | Ax = b, x \geq 0 \right\} \tag{7}$$

The initial task is solved to get the initial first-stage solution. The initial point can also be selected in another way. Our program allows to select from the several initial point alternatives. If the initial task is unlimited and the final solution x^* of this problem does not exist, then, we use our modification:

$$\min_{x,y} \left[c^T x + q^T y u \right], \tag{8}$$

where

$$\begin{aligned} &Ax = b, \\ &Tx + Wy = h, \\ &x \geq 0,\ y_0 \geq 0,\ u << 1. \end{aligned} \tag{9}$$

In this case, we include the second-stage objective function vector and constraints. The second-stage objective function vector is multiplied by a very small value u.

In other cases the main linear task (master) is solved:

$$\min_{x,\theta} \left[c^T x + \theta \right], \tag{10}$$

where

$$Ax = b,\ \ x \geq 0,\ \ \theta \in R, \tag{11}$$

$$D_l x \geq d_l,\ \ l = \overline{1, r}, \tag{12}$$

$$E_l x + \theta \geq e_l,\ \ l = \overline{1, s}. \tag{13}$$

If the final solution x^* of this problem does not exist, then a modified task is solved (see Equations (8) and (9)), where u is a very small value.

Step 2.

For all scenarios $k = \overline{1, K}$, the linear task (see Equations (14), (15) and (16)) is solved:

$$\phi' = \min_{y, \upsilon^+, \upsilon^-} \left[e^T \upsilon^+ + e^T \upsilon^- \right], \tag{14}$$

where

$$Wy + I\upsilon^+ - I\upsilon^- = h_k - T_k x^\nu, \tag{15}$$

$$y \geq 0,\ \ \upsilon^+ \geq 0,\ \ \upsilon^- \geq 0,\ \ e^T = (1, \ldots, 1). \tag{16}$$

For all k, where $\phi' > 0$, we define the feasible cut variables:

$$D_{r+1} = \left(\sigma^\nu\right)^T T_k\ ,\ \ d_{r+1} = \left(\sigma^\nu\right)^T h_k, \tag{17}$$

where σ^ν is a simplex multiplier.

Afterwards, we add a feasible cut constraint (see Equation (12)) and go to Step 1.

In other cases, if for all k, $\phi' = 0$, we go to Step 3.

Step 3.

The second-stage tasks (subproblems) are solved for all scenarios $k = \overline{1, K}$:

$$\omega = \min_y \left\{ q_k^T y | Wy = h_k - T_k x^\nu, y \geq 0 \right\} \tag{18}$$

Next, variables for the optimality cut are defined:

$$E_{s+1} = \sum_{k=1}^{K} p_k \left(\pi_k^\nu\right)^T T_k, \tag{19}$$

$$e_{s+1} = \sum_{k=1}^{K} p_k \left(\pi_k^\nu\right)^T h_k, \tag{20}$$

$$\omega^{\nu} = e_{s+1} - E_{s+1}. \quad (21)$$

If $\theta^{\nu} \geq \omega^{\nu}$, stop, x^{ν} is the optimal solution vector.

Otherwise, add optimality cut constraint (see Equation (13)) and go to Step 1.

End of the algorithm.

Thus, in the first iteration, the initial solution x^* of task (see Equation (7)) is obtained. If the finite solution x^* does not exist, then, the modified task (see Equations (8) and (9)) is solved, where the second-stage variables are included in the objective function by multiplying a very small coefficient u.

The second-stage task is solved for all scenarios $k = \overline{1,K}$ until all the objective function values are zero.

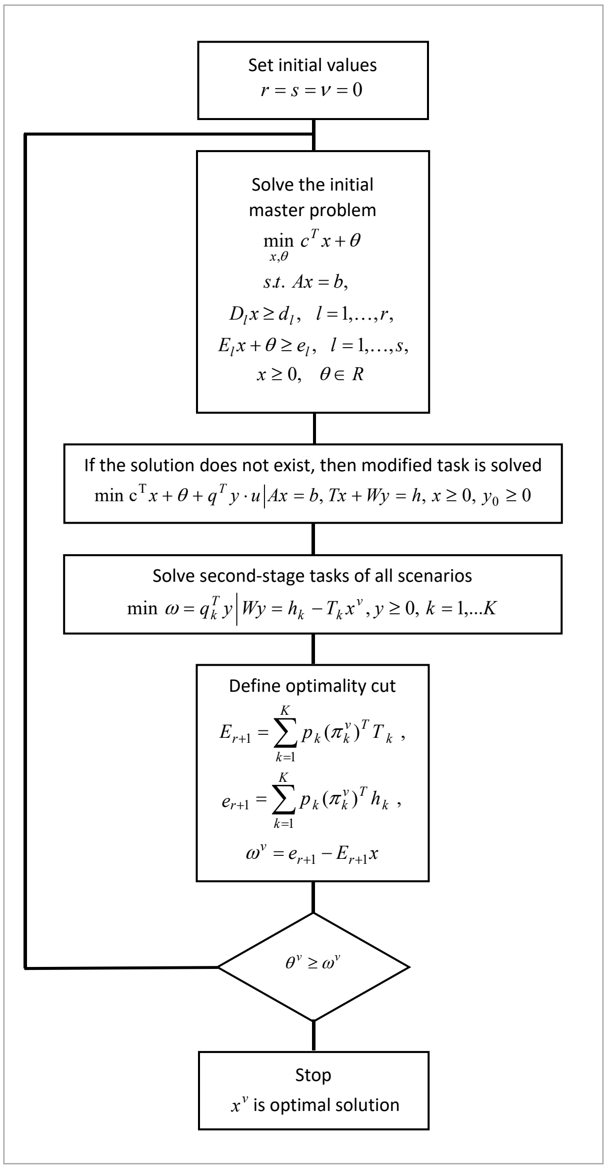
The algorithm scheme is shown in Fig. 1.

The algorithm is based on the decomposition method described by Bierge et al. [1]. However, several changes have been made.

Before the first iteration, the initial values are defined. Next, the main linear task (master) is solved. If the final solution of this problem does not exist, then, a modified task is solved. Next, the second-stage tasks (subproblem) are solved for all scenarios and variables for the optimal cut are defined. Afterwards, the stop condition is checked, and if it is not satisfied, iterations are repeated. The iterations are repeated until the required accuracy is obtained.

Figure 1

A modified L-shaped algorithm


Set initial values
$r = s = \nu = 0$
Solve the initial master problem
$\min_{x,\theta} c^T x + \theta$
$s.t.\ Ax = b,$
$D_l x \geq d_l,\ l = 1,\ldots,r,$
$E_l x + \theta \geq e_l,\ l = 1,\ldots,s,$
$x \geq 0,\ \theta \in R$
If the solution does not exist, then modified task is solved
$\min c^T x + \theta + q^T y \cdot u \,|\, Ax = b, Tx + Wy = h, x \geq 0, y_0 \geq 0$
Solve second-stage tasks of all scenarios
$\min \omega = q_k^T y \,|\, Wy = h_k - T_k x^{\nu}, y \geq 0, k = 1,\ldots K$
Define optimality cut
$E_{r+1} = \sum_{k=1}^{K} p_k (\pi_k^{\nu})^T T_k,$
$e_{r+1} = \sum_{k=1}^{K} p_k (\pi_k^{\nu})^T h_k,$
$\omega^{\nu} = e_{r+1} - E_{r+1} x$
$\theta^{\nu} \geq \omega^{\nu}$
Stop
x^{ν} is optimal solution

4. Scenario Aggregation Method

The scenario aggregation method was proposed by Rockafeller and Wets [10]. Later, Wets described the main aggregation principles in the scenario analysis and stochastic optimization [19]. Further, this method has been applied in solving various specific problems. In Jönsson et al. [6], the aggregation technique is applied to the two-stage production task when it is necessary to distribute the given budget for the purchase of product components. In many stages of uncertainty, optimization problems have been analysed for scenarios [11].

Subsequently, the scenario aggregation method was analysed in Cambou et al. [3], and the main criteria for applying this method were formulated, based on examples. The essence of the scenario aggregation method is described below.

The idea is that, when dealing with various subproblems and their optimal solutions, one can discover similarities and trends, and ultimately get a well-insured solution to the underlying problem [19]. The general approach in practice is based on the scenario analysis [6, 19].

If we have a stochastic optimization task described in a probability space (Ψ, P), where Ψ is a set of possible realizations, and P is the associated probability distribution, the problem is:

$$\min_x E\{g(x,\xi)\} = \int g(x,\xi)\,dP(\xi), \tag{22}$$

where $x \in D \subset R^n$, D is a set of feasible solutions determined by constraints, and g is the criterion function.

Since the model depends on the random vector ξ and its probabilistic distribution P, it cannot be used as an appropriate simulation tool when we have only limited information on the distribution of random parameters. In such cases, the scenario analysis is used most commonly [19].

In case the uncertainty is modelled by a few scenarios $s^k, k = \overline{1,K}$, and for each scenario s^k, one finds the solution of subproblem P_k:

$$\min_x \left\{ g(x,s) | x \in D_k \subset R^n \right\}. \tag{23}$$

Suppose we know how to get the solution to each individual scenario taken. The problem is how to deal with different *s*-dependent vectors in solving the problem of combining them and obtaining a common solution [11].

Assuming that the optimal solution exists for all scenarios $s^k, k = \overline{1,K}$, the optimal solution is:

$$x^k \in arg \min_x \left\{ g(x,s) | x \in D_k \right\}. \tag{24}$$

When the solution to each scenario is computed, they are analysed in order to find common trends or solution clusters and determine how the solution would be calculated, if the scenario s' really appears. The average solution is calculated multiplying all solutions x^k by the probability of scenarios, and the average solutions are analysed further. The ultimate goal of the analysis is to get one solution that can be used to make a decision.

Constructing an estimate indicating the average solution:

$$\hat{x} = \sum_{k=1}^{K} p_k x^k, \tag{25}$$

where x^k is the solution of the scenario s^k, $k = \overline{1,K}$, and p_k are probabilities (weights) of the scenario. These probabilities are necessarily nonnegative, and up to 1 [19]. The solution $\hat{x}$ does not depend on the scenario, i.e., in general, it will respond more to all the most likely events possible than specific solutions of the individual scenario solution x^k. However, $\hat{x}$ is not necessarily possible. The solution x is acceptable, if it is possible for each particular scenario, i.e., $x \in D_k$ for all $s^k, k = \overline{1,K}$.

A stochastic optimization problem is defined below in which each scenario s^k relates to the probability of the scenario p_k:

$$\min_x \sum_{k=1}^{K} \left(p_k g\left(x, s^k\right) \right), \tag{26}$$

where $x \in \bigcap_{s^k} D_k$.

The optimal solution of the problem is x^*.

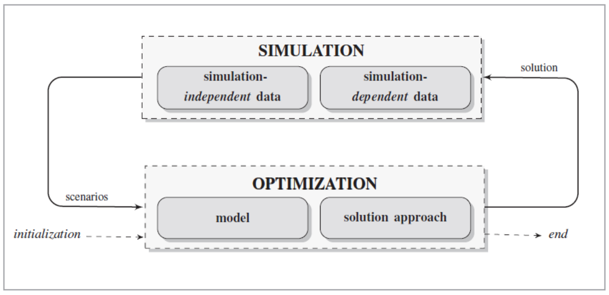
5. Stochastic Simulation

Simulation is mainly used in the following cases: where it is impossible or very expensive to get data from a certain real process; where the system investigated is very complicated and cannot be described by mathematical equations having analytical solutions; where the system is described in a mathematical model, but it is impossible to get a solution by using analytical techniques; where it is impossible or very expensive to perform experiments of the mathematical system model.

Simulation optimization coupling [2], illustrated in Fig. 2, is an active area in the field of stochastic programming. In this connection, the simulation is generally used to generate scenarios in accordance with the probability distributions data [2].

Statistical modelling uses a selection of stochastic random variables, and is defined as experimentation with the model in time. Random values were simulated by generating Gaussian values. The Normal or Gaussian law describes the distribution of such a random size obtained, by summing up a large number of other independent random variables that are not dominant among them.

Figure 2
Simulation optimization coupling [2]



The normal law is very often applied in practice. It is an idealized mathematical model for analysing data that are roughly normal. Therefore, in this work, random scenarios were generated according to the Gaussian law.

6. Optimization Using IBM ILOG CPLEX

For solving optimization problems we used the IBM ILOG CPLEX Optimization Studio that was integrated with Microsoft Visual Studio.

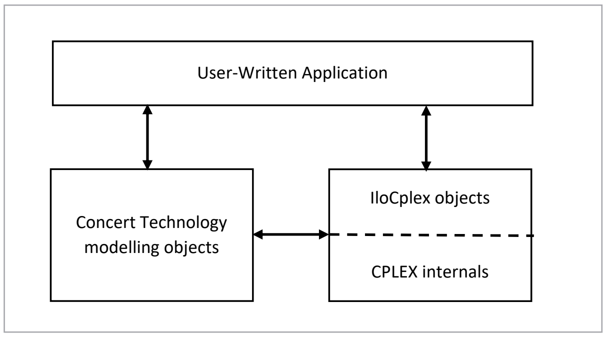
The IBM ILOG CPLEX Optimization Studio is an optimization software package [5]. It enables a rapid development and deployment of decision optimization models using mathematical and constraint programming [5]. The efficiency of solution of a task depends on the CPLEX integration. Therefore, it is very important to properly select and match all the parameters of the task.

IBM ILOG CPLEX offers C, C++, Java, .NET, and Python libraries that solve linear programming and related problems [5]. Specifically, it solves linearly or quadratically constrained optimization problems, where the objective to be optimized can be expressed as a linear function or a convex quadratic function [5]. The variables in the model may be declared as continuous or further constrained to take only integer [5].

The program using the CPLEX Concert Technology to solve optimization problems is shown in Fig. 3.

The optimization part of the user's application program is captured in a set of interacting C++ modelling and solving objects that the application creates and controls [5]. Modelling objects are used to define the optimization problem. Solving objects in the instance

Figure 3
A view of Concert Technology for C++ users [5]



of *IloCplex* are used to solve models created by the modelling objects.

An *IloCplex* object reads a model, extracts its data, solves the problem, and answers queries on solution.

The Concert Technology model consists of a set of C++ objects. Each variable, each constraint, each special ordered set (SOS), and the objective function in the model are all represented by objects of the appropriate Concert Technology class.

The environment is the first object created in an application. The environment object needs to be available to the constructor of all other Concert Technology classes. After creating the environment, a Concert application is ready to create one or more optimization models. After an *IloModel* object has been constructed, it can be populated with modelling variables, constraints, and objective function. The class *IloCplex* solves a model. Query methods access information about the solution. CPLEX supports reading models from files and writing models to files in several languages (e.g., LP format, MPS format).

We describe above the CPLEX model creation and the solving process used in our research.

First, we define the vector of solution variables as *IloNumVarArray*.

Next, we define the first-stage and the second-stage objective function vectors as *IloExpr*, and add them to the model using *model.add()* method.

Now, we define the matrixes describing the left-hand of the first- and the second-stage constraints as *IloExprArray* .

Next we define the first- and the second-stage constraints as *IloRangeArray*, and add them to the model using *model.add()* method.

Finally, we define created model as *IloCplex* model and solve it using the CPLEX solver method *solve()*.

Next, we can apply the information of the solution to our further calculations. The main information of the solution are as follows:

- *cplex.getStatus()* gets the status of the solution;
- *cplex.getObjValue()* gets the value of the objective function;
- *cplex.getValues(vals, vars)* gets the values of the solution.

7. Investigation of the Proposed Method

The scenario aggregation method was applied for solving a two-stage SLP task. The calculations were performed by a computer, which parameters are: Intel(R) Core(TM) i7-4500U CPU @ 1,80 GHz 2.4 GHz, 8.00GB, x64-based processor. The program was implemented in the Microsoft Visual Studio 2012 C++ language, using the IBM ILOG CPLEX optimization package. The first two-stage task has 20 variables and 10 constraints in the first-stage, and 30 variables and 20 constraints in the second (20x30 task). The second two-stage task has 60 variables and 30 constraints in the first-stage, and 90 variables and 60 constraints in the second (60x90 task).

In the first case, the 2000 normally distributed scenarios were divided into 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 and 2000 groups. The calculations (2000 scenarios) were repeated 100 times for each case of division.

The averaged results for all cases of division are presented in Table 1 (average result = result of 2000 scenarios repeated 100 times divided by 100).

The average value of the main (master) task objective function is $TF = 182.12$.

In the second case the 1800 normally distributed scenarios were divided into 1, 2, 5, 10, 20, 50, 100, 200, 600, 900 and 1800 groups. The calculations (1800 scenarios) were repeated 100 times for each case of division.

The averaged results for all cases of division are presented in Table 2 (average result = result of 1800 scenarios repeated 100 times divided by 100).

Table 1
Averaged results of calculation (20x30 task)

Number of groups	Time (min)	Number of iterations
1	22.2	34.3
2	20.0	30.2
5	17.1	26.3
10	15.3	23.1
20	13.0	20.2
50	11.0	17.1
100	9.3	14.2
200	7.8	12.1
500	5.7	8.7
1000	4.7	7.1
2000	4.1	6.1

The average value of the main (master) task objective function is $TF = 299.07$.

The results indicate that by increasing the number of groups, the computation time and the number of iterations required to achieve the optimal value, de-

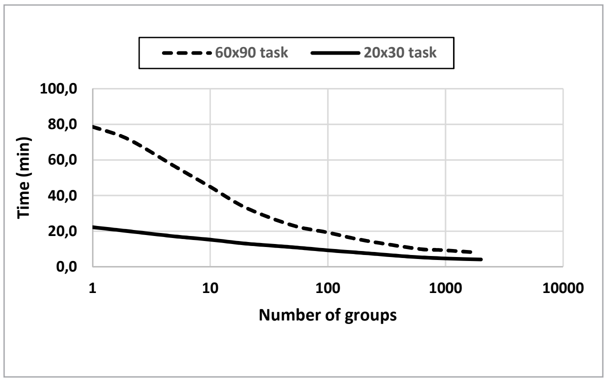
Table 2
Averaged results of calculation (60x90 task)

Number of groups	Time (min)	Number of iterations
1	78.6	96.1
2	71.7	87.3
5	56.4	68.2
10	44.9	54.3
20	33.2	40.7
50	23.3	28.3
100	19.3	23.7
200	14.9	18.2
600	10.0	12.1
900	9.4	11.1
1800	8.1	9.2

crease. Dependence of time on the number of groups is shown in Fig. 4.

Figure 4

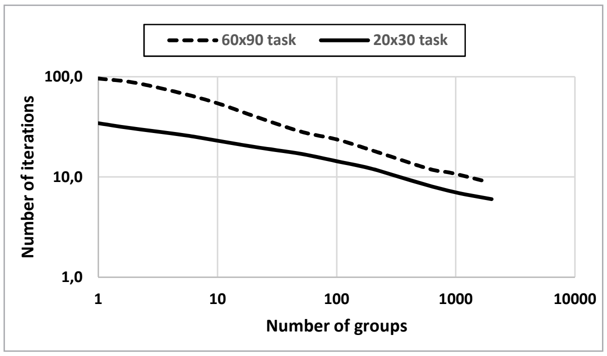
Dependence of time (y axis) on the number of groups (x axis)



Dependence of the number of iterations on the number of groups is shown in Fig. 5.

Figure 5

Dependence of the number of iterations (y axis) on the number of groups (x axis)



Dependence of the average time of one iteration calculation on the number of groups is shown in Fig. 6 (20x30 task) and Fig. 7 (60x90 task).

As we can see, the optimal time of one iteration calculation was obtained when the number of groups is between 100 and 200, i.e. one group have contain about 10–20 scenarios. When the number of groups increase, the calculation time of the one iteration increases too, because the number of constraints of the main (master) task increases respectively.

When we use our proposed two-stage SLP problem decomposition algorithm modification, we must ensure the solution existence for every scenario. For

Figure 6

Dependence of the average time of one iteration calculation (y axis) on the number of groups (x axis) (20x30 task)

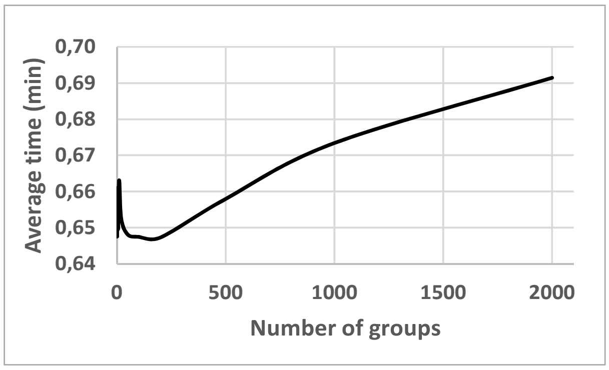
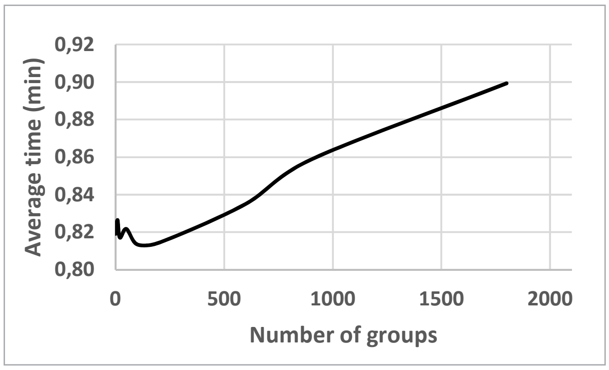


Figure 7

Dependence of the average time of one iteration calculation (y axis) on the number of groups (x axis) (60x90 task)



that purpose, the values of parameter u related to solution existence were investigated.

At the beginning, the parameter value u was set to $u = 1$. Then it was reduced to 0.0001, and the problem was solved. The process was carried out as long as the solution existed.

The calculation results showed, that the minimal u value which can ensure solution existence solving such tasks, is $u = 0.001$.

8. Conclusion

A stochastic optimization method has been proposed to solve two-stage SLP problems, where first-stage task is unbounded. The method consists of decomposition, aggregation and solving technologies coupling.

The decomposition method, based on Sample Average Approximation for solving SLP problems, is presented, where the stochastic variables are described by an absolutely continuous probability law.

The investigation result obtained on choice of parameter u of our proposed two-stage SLP problem decomposition method modification algorithm shows that minimal value, which ensure solution existence, is $u = 0.001$.

The efficient CPLEX integration has been proposed for solving SLP problems by the L-shaped algorithm which enables us to adapt integration parameters to specific problems.

The obtained results show that the number of iterations decreases by increasing the number of scenario groups (see Fig. 5), concurrently the overall calculation time decreases for the same accuracy (see Fig. 4).

The aggregation of the scenarios into 100-200 groups gives the best average execution time per iteration solving the SLP by modified Lshaped algorithm, if the following conditions are met: the number of variables reaches up to 100; random scenarios are generated according to the Gaussian law; number of the scenarios is about 2000. Increasing the number of groups, the average calculation time of one iteration increases as well (see Figs. 6 and 7). This is due to the fact that if the number of groups increases, the number of constraints of the main (master) task increases, respectively.

The disadvantage of the method is that the different classes of problems can correspond to different numbers of optimal groups. If we do not know the number of optimal groups, our choice may be irrational.

References

1. Birge, J. R., Louveauxl, F. V. Introduction to Stochastic Programming. Springer, New York, 2011. hhttps://doi.org/10.1007/978-1-4614-0237-4
2. Borodin, V., Bourtembourg, J., Hnaien, F., Labadie, N. COTS Software Integration for Simulation Optimization Coupling: Case of ARENA and CPLEX Products. Working Paper EMSE CMP–SFL, 2018. https://hal-emse.ccsd.cnrs.fr/emse-01687555/document. Accessed on January 27, 2018.
3. Cambou, M., Filipović, D. Model Uncertainty and Scenario Aggregation. Mathematical Finance, Forthcoming, Swiss Finance Institute Research Paper No. 14-38. February 2, 2014. https://doi.org/10.2139/ssrn.2441328
4. Fountoulakis, K., Gondzio, J. Performance of First- and Second-Order Methods for L1-Regularized Least Squares Problems. Computational Optimization and Applications, 2016, 65(3), 605-635. https://doi.org/10.1007/s10589-016-9853-x
5. IBM Inc. IBM ILOG CPLEX Optimization Studio Getting Started with CPLEX (Version 12 Release 6). IBM Corporation, 2014.
6. Jönsson, H., Jörnsten, K., Silver, E. A. Application of the Scenario Aggregation Approach to a Two-stage, Stochastic, Common Component, Inventory Problem with a Budget Constraint. European Journal of Operational Research, 1993, 68(2), 196-211. https://doi.org/10.1016/0377-2217(93)90303-5
7. King, A. J., Wallace, S. W. Modeling with Stochastic Programming. Springer Series in Operations Research and Financial Engineering, Springer, New York, 2012. ISBN 978-0-387-87817-1.
8. Ogbe, E., Li, X. A New Cross Decomposition Method for Stochastic Mixed-Integer Linear Programming. European Journal of Operational Research, 2017, 256(2), 487-499. https://doi.org/10.1016/j.ejor.2016.08.005
9. Rahmaniani, R., Crainic, T. G., Gendreau, M., Rei, W. The Benders Decomposition Algorithm: A Literature Review. European Journal of Operational Research, 2017, 259(3), 801-817. https://doi.org/10.1016/j.ejor.2016.12.005
10. Rockafellar, R. T., Wets, R. J.-B. Scenarios and Policy Aggregation in Optimization Under Uncertainty. IIASA Working Paper. IIASA, Laxenburg, Austria: WP-87-119, 1987. http://pure.iiasa.ac.at/id/eprint/2933/1/WP-87-119.pdf. Accessed on June 25, 2017.
11. Rockafellar, R. T., Wets, R. J.-B. Scenarios and Policy Aggregation in Optimization under Uncertainty. Mathematics of Operations Research, 1991, 16(1), 119-147. https://doi.org/10.1287/moor.16.1.119
12. Sakalauskas, L. Application of the Monte-Carlo Method to Nonlinear Stochastic Optimization with Linear Constraints. Informatica, 2004, 15(2), 271-282. ISSN 0868-4952.
13. Schulzea, T., Grothey, A., McKinnon, K. A Stabilised Scenario Decomposition Algorithm Applied to Stochastic Unit Commitment Problems. European Journal of Operational Research, 2017, 261(1), 247-259. https://doi.org/10.1016/j.ejor.2017.02.005

14. Shabbir, A., Shapiro, A. The Sample Average Approximation Method for Stochastic Programs with Integer Recourse. ISyE Technical Report, 2002. https://www2.isye.gatech.edu/~sahmed/saasip.pdf. Accessed on June 21, 2017.

15. Sun, L., Liu, W., Chai, T., Wang, H., Zheng, B. Crane Scheduling of Steel-Making and Continuous Casting Process Using the Mixed-Timed Petri Net Modelling via CPLEX Optimization. IFAC Proceedings Volumes, 2011, 44(1), 9482-9487. https://doi.org/10.3182/20110828-6-IT-1002.00170

16. Ušpurienė, A., Sakalauskas, L., Giuliani, S., Meloni, C. ABC Model for Economic Development of a Firm. Technological and Economic Development of Economy, 2016, 22(4), 512-531. https://doi.org/10.3846/20294913.2016.1190420

17. Vamanan, M., Wang, Q., Batta, R., Szczerba, R. J. Integration of COTS Software Products ARENA & CPLEX for an Inventory/Logistics Problem. Computers & Operations Research, 2004, 31(4), 533-547. https://doi.org/10.1016/S0305-0548(03)00010-8

18. Wang, T., Wang, S., Meng, Q. Liner Ship Fleet Planning: Models and Algorithms, Part Two: Mathematical Modeling, Elsevier, 2017, 61-72. ISBN: 978-0-12-811502-2.

19. Wets, R. J.-B. The Aggregation Principle in Scenario Analysis and Stochastic Optimization. Algorithms and Modelling in Mathematical Programming, Springer-Verlag, Berlin, Heidelberg, 1989, 91-113. https://doi.org/10.1007/978-3-642-83724-1_4

20. Yakowitz, D. S. Two-Stage Stochastic Linear Programming: Stochastic Decomposition Approaches. The University of Arizona, Dissertation, 1991.