A Hybrid Benders' Decomposition Method for Solving Stochastic Constraint Programs with Linear Recourse

S. Armagan Tarim[1] and Ian Miguel[2]

[1] University College Cork, Cork Constraint Computation Centre, Cork, Ireland,
at@4c.ucc.ie, http://www-users.cs.york.ac.uk/~at/
[2] University of St.Andrews, School of Computer Science, St.Andrews, Scotland,
ianm@dcs.st-and.ac.uk, http://www-users.cs.york.ac.uk/~ianm/

Abstract. We adopt Benders' decomposition algorithm to solve scenario-based Stochastic Constraint Programs (SCPs) with linear recourse. Rather than attempting to solve SCPs via a monolithic model, we show that one can iteratively solve a collection of smaller sub-problems and arrive at a solution to the entire problem. In this approach, decision variables corresponding to the initial stage and linear recourse actions are grouped into two sub-problems. The sub-problem corresponding to the recourse action further decomposes into independent problems, each of which is a representation of a single scenario. Our computational experience on stochastic versions of the well-known template design and warehouse location problems shows that, for linear recourse SCPs, Benders' decomposition algorithm provides a very efficient solution method.

1 Introduction

Stochastic constraint programming (*SCP*, see [12, 15]) extends constraint programming to deal with both *decision* variables, which can be set by the decision-maker, and *stochastic* variables, which follow some discrete probability distribution function. This framework is designed to model a wide variety of decision problems involving uncertainty and probability. Examples include nurse rostering given an uncertain workload and constructing a balanced bond portfolio.

Tarim *et al.* [15] provide a semantics for stochastic constraint programs based on *scenarios*, where a scenario is a possible set of values for the stochastic variables. Based on this semantics, they compile stochastic constraint programs down into conventional (non-stochastic) constraint programs. The advantage of this compilation is that existing constraint solvers can be used without modification. However, the number of scenarios grows exponentially with the number of decision stages, where each stage consists of a set of decision variables and a set of stochastic variables whose combined assignments determine the structure of the next stage. Tarim *et al.* propose a number of scenario reduction algorithms to reduce the scenario tree considered. These algorithms determine a subset of scenarios and a redistribution of probabilities relative to the preserved scenarios. Generally, however, this approach yields sub-optimal solutions.

This paper presents Benders' decomposition (*BD*) algorithm as an optimisation method for stochastic constraint programs with linear recourse (*SCPwLR*). *SCPwLR* constitutes an important subgroup of stochastic constraint programs. In this type of SCP the initial stage decisions involve discrete variables, whereas the following recourse actions comprise only continuous decision variables. A typical example of *SCPwLR* is the Warehouse Location Problem (WLP) (see [10], [11] and [13]) with stochastic demand. In Section 5, the capacitated version of the WLP with stochastic demand is addressed, along with a stochastic version of the Template Design problem [14], and the computational performance of Benders' algorithm is investigated.

The paper is organised as follows. Section 2 provides background, including an overview of Benders decomposition. Section 3 introduces the linear recourse stochastic constraint programs and an application of BD to such programs. Section 4 gives an illustrative example of BD applied to *SCPwLR* by means of the classical news vendor problem. In the following section, the computational efficiency of BD is investigated. Section 6 concludes the paper and points out important future work.

2 Background

This section gives the necessary background detail in stochastic constraint programming and Benders decomposition. We begin with the former.

2.1 Stochastic Constraint Programming

A stochastic constraint satisfaction problem [12] consists of a 6-tuple $\langle \mathcal{X}, \mathcal{S}, \mathcal{D}, \mathcal{P}, \mathcal{C}, \theta \rangle$. $\mathcal{X}$ is a set of decision variables, and $\mathcal{S}$ is a set of stochastic variables. $\mathcal{D}$ is a function mapping each element of $\mathcal{X}$ and each element of $\mathcal{S}$ to a domain of potential values. A decision variable $x \in \mathcal{X}$ is *assigned* a value from its domain. $\mathcal{P}$ is a function mapping each element of $\mathcal{S}$ to a probability distribution for its associated domain. $\mathcal{C}$ is a set of constraints, where a constraint $c \in \mathcal{C}$ on variables $x_i, \ldots, x_j$ specifies a subset of the Cartesian product $\mathcal{D}(x_i) \times \ldots \times \mathcal{D}(x_j)$ indicating mutually-compatible variable assignments. The subset of $\mathcal{C}$ that constrain at least one variable in $\mathcal{S}$ are *chance* constraints. θ is a function mapping each chance constraint to the interval [0,1], indicating the fraction of scenarios in which the constraint must be satisfied. Note that a chance constraint with a threshold of 1 is equivalent to a hard constraint.

A stochastic CSP consists of a number of *decision stages*. In a one-stage stochastic CSP, the decision variables are set before the stochastic variables. In an n-stage stochastic CSP, $\mathcal{X}$ and $\mathcal{S}$ are partitioned into n disjoint sets, $\mathcal{X}_1, \ldots, \mathcal{X}_n$ and $\mathcal{S}_1, \ldots, \mathcal{S}_n$. To solve an n-stage stochastic CSP an assignment to the variables in $\mathcal{X}_1$ must be found such that, given random values for $\mathcal{S}_1$, assignments can be found for $\mathcal{X}_2$ such that, given random values for $\mathcal{S}_2$, ..., assignments can be found for $\mathcal{X}_n$ so that, given random values for $\mathcal{S}_n$, the hard constraints are satisfied and the chance constraints are satisfied in the specified

fraction of scenarios. As noted in the introduction, the *SCPwLR* is a 2-stage stochastic CSP in which the domains of $\mathcal{X}_1$ are discrete, but the domains of $\mathcal{X}_2$ are continuous. Furthermore, the constraints on $\mathcal{X}_2$ are all linear.

2.2 Benders Decomposition

Although Benders' decomposition algorithm dates back to the 1960s and there is now a sizeable OR literature in this area extending the original approach, it has only recently been used by the constraint programming community in developing hybrid models. The reader is directed to Benoist et al. [3], Xia et al. [17], Eremin and Wallace [4], Hooker and Ottosson [7], Thorsteinsson [16], Jain and Grossmann [9] for applications of BD in constraint programming.

Benders decomposition [2, 5] was presented for solving models of the type:

$$\max\{c^T x + f(y) | Ax + F(y) \leq b, x \in \mathbb{R}^p_+, y \in S\} \tag{1}$$

where $x \in \mathbb{R}^p_+$ (the p-dimensional non-negative Euclidean space), $y \in \mathbb{R}^q$, and S is an arbitrary subset of $\mathbb{R}^q$. Furthermore, A is an (m, p) matrix, $f(y)$ is a scalar function and $F(y)$ an m-component vector function both defined on S, and b and c are fixed vectors in $\mathbb{R}^m$ and $\mathbb{R}^p$, respectively.

A key concept in Benders' algorithm is that of partitioning the variables into two sets – x and y – and projecting the problem onto the complicating variables, y. Benders' method decomposes this model in such a way that it can be solved as an alternating sequence of linear programs and programs of "complicating" variables. In the case of Eq.(1), once y is fixed to $\bar{y}$, the initial linear program is:

$$f(\bar{y}) + \min\{(b - F(\bar{y}))\lambda | A^T \lambda \geq c, \lambda \in \mathbb{R}^m_+\}, \tag{2}$$

In other words, the algorithm partitions the given problem in Eq.(1) into two such subproblems: a programming problem (which may be linear, non-linear, discrete, etc.) defined on S, and a linear programming problem defined in $\mathbb{R}^p_+$. An example is the mixed-integer programming problem in which certain variables may assume any value on a given interval, whereas others are restricted to integral values only. Then, in order to avoid the laborious calculation of a complete set of constraints for the feasible region in the first problem, a multi-step procedure is designed leading, in a finite number of steps, to a set of constraints determining an optimum solution of the given problem.

The classic BD algorithm was proposed for mixed-integer linear programming problems, the cut generation of which is based on the duality theorem of linear programming. The algorithm functions as follows: It determines trial values for the addressed problem by solving a program called the master problem – the program of complicating variables. The cost of this trial plan is determined using the so-called slave problem, Eq.(2). The slave problem also calculates dual multipliers, λ, which measure the marginal change in the trial plan. These dual multipliers are used to form new constraints that are added to the master problem, which is then re-solved to determine a new trial plan. The process continues alternately solving the master and slave problems, until the algorithm has found an optimal plan or one that is within an acceptable tolerance of optimality.

3 Benders Decomposition Applied to *SCPwLR*

In this section, BD is proposed as a solution method for *SCPwLR*. We consider only the 2-stage *SCPwLR*, but the method can easily be extended to address multi-stage *SCPwLR* without loss of generality. The method described is a *hybrid* since it requires a collaboration of CP and LP methods.

As mentioned, BD partitions the decision variables into two sets, x and y. For *SCPwLR*, partitioning is made with respect to the two decision stages: the first stage decision variables, which constitute the set of complicating variables y, form a CP model, and, since the recourse action is assumed to be linear, the second stage decision variables form an LP model. Hence, the "master" problem is a CP, whereas the "slave" problem, corresponding to the scenarios and the recourse actions taken, is an LP. In this context, Benders decomposition achieves separability of the second stage decisions, solving a separate LP for each scenario.

Consider the following 2-stage *SCPwLR*: $\min\ \{f(y)+\sum_{k=1}^{K} p_k Q_k(y) | y \in Y\}$. Here, k indexes the finitely-many scenarios, with p_k the probability of scenario k. The first-stage variables y are set before the scenario is observed. After the kth scenario is observed, the set of second-stage decision variables x_k are set. The cost (assumed to be linear) of the second stage in scenario k is $Q_k(y) = \min\{q_k x | W_k x = h_k - T_k y, x \geq 0\}$. That is, x is a recourse, which must be chosen so as to satisfy some linear constraints in the least costly way.

We assume that recourse is *complete*, i.e., for any choice of y and scenario, there is always a non-empty set of x, $\{x | W_k x = h_k - T_k y, x \geq 0\} \neq \emptyset$. The objective is to minimize the expected total costs of both stages.

The deterministic equivalent model is a large-scale problem, which simultaneously selects the first-stage variables y and the second-stage variables x_k for every scenario k.
SCPwLR Model:

$$z = \min\ \{f(y) + \sum_{k=1}^{K} p_k q_k x_k \ \mid\ T_k y + W_k x_k = h_k, \quad x_k \geq 0, \quad y \in Y\} \tag{3}$$

3.1 Independent subproblems

Given an arbitrary first stage decision $\overline{y}$, define a function $Q_k(\overline{y})$ equal to the optimum of the second stage for each scenario $k = 1, ..., K$:
Slave (*Primal*) *Model*:

$$Q_k(\overline{y}) = \min\ \{q_k x_k \ | W_k x_k = h_k - T_k \overline{y}, \quad x_k \geq 0\} \tag{4}$$

Now, an upper bound on the optimal value of z, defined in (3), is:

$$f(\overline{y}) + \sum_{k=1}^{K} p_k Q_k(\overline{y}) \tag{5}$$

Under the assumption of complete recourse, the linear programming dual of the second-stage problem for scenario k, as given in (4), is the linear program:
Slave (*Dual*) *Model*:

$$Q_k(\overline{y}) = \max \ \{(h_k - T_k\overline{y})\lambda_k \ \mid \ W_k^T\lambda_k \leq q_k, \quad \lambda_k \ \text{free}\} \tag{6}$$

Note that the constraints are now independent of y; in other words, the feasible region is not affected by the choice of y. Denote by $\Lambda_k = \{\lambda_k | W_k^T\lambda_k \leq q_k\}$ the polyhedral feasible region of the second-stage problem for scenario k. Denote by $\widehat{\lambda}_k^i$ the ith extreme point of Λ_k, $i = 1, ..., I_k$, where I_k is the total number of extreme points of the problem for scenario k. By enumerating the large, but finite, number of extreme points of Λ_k, we can write, $Q_k(y) = \max_{i=1,...,I_k}\{\widehat{\lambda}_k^i(h_k - T_k y)\}$, which demonstrates that $Q_k(y)$ is a piecewise-linear convex function.

3.2 Complete and Partial Master Problems

Benders' "complete master problem" then uses this representation of $Q_k(y)$ to provide an alternative method for evaluating z,
Complete Master Model:

$$z = \min \ \{f(y) + \sum_{k=1}^{K} p_k \max_{i=1,...,I_k}\{\widehat{\lambda}_k^i(h_k - T_k y)\} \ \mid \ y \in Y\} \tag{7}$$

While it is possible in principle to solve the problem using Benders' complete master problem, in practice the magnitude of the number of dual extreme points makes it prohibitively expensive. However, if a subset of the dual extreme points of Λ_k are available then we obtain an underestimate of $Q_k(y)$, which we denote:

$$Q_k^{'}(y) = \max_{i=1,...,M_k}\{\widehat{\lambda}_k^i(h_k - T_k y)\} \tag{8}$$

where $M_k \leq I_k$.

Using dual information obtained after M evaluations of $Q_k(y)$, we obtain a "partial master problem", which provides a lower bound on the solution of z:
Partial Master Model:

$$\min \ \{f(y) + \sum_{k=1}^{K} p_k \max_{i=1,...,M_k}\{\widehat{\lambda}_k^i(h_k - T_k y)\} \mid y \in Y\} \tag{9}$$

However, there is no guarantee that partial master problem yields a bounded solution. If it produces an unbounded solution then the direction of the *extreme ray* must be determined and the Benders cut, $0 \geq \widehat{\lambda}_k^r(h_k - T_k y)$, must be added accordingly to bound the unbounded polyhedral set.

Benders' algorithm solves the current "partial master problem", obtaining a new $\overline{y}$ (a new trial solution) and, an underestimate $\sum_{k=1}^{K} p_k Q_k^{'}(\overline{y})$ of the associated expected second-stage cost.

Algorithm 1: BD-SCPwLR

input : Set of scenarios, S
$z = \min\{f(y) + \sum_{\forall k \in S} p_k q_k x_k | T_k y + W_k x_k = h_k, x_k \geq 0, y \in Y\}$
output: $\{z^*, y^*, x_k^*\}$
begin
  $\bar{y} \leftarrow$ an initial feasible y
  $c \leftarrow f(\bar{y})$
  $up \leftarrow \infty$
  $low \leftarrow -\infty$
  $\forall k \in S,\ Cut_k \leftarrow \emptyset$
  while $up - low > \epsilon$ **do**
    for $k \in S$ **do**
      $Slave_k \leftarrow Q_k(\bar{y}) = \max\{(h_k - T_k\bar{y})\lambda_k | W_k^T \lambda_k \leq q_k, \lambda_k \text{ free}\}$
      $\widehat{\lambda}_k \leftarrow Solve(Slave_k)$
      $Cut_k \leftarrow Cut_k \cup \{\widehat{\lambda}_k(h_k - T_k y)\}$
    $c \leftarrow f(\bar{y}) + \sum_{\forall k \in S} p_k Q_k(\bar{y})$
    if $c < up$ **then** $up \leftarrow c$
    $Master \leftarrow \min\{f(y) + \sum_{\forall k \in S} p_k \max\{Cut_k\} | y \in Y\}$
    $\bar{y} \leftarrow Solve(Master)$
    $c \leftarrow \min\{f(\bar{y}) + \sum_{\forall k \in S} p_k \max\{Cut_k | y = \bar{y}\}\}$
    if $c > low$ **then** $low \leftarrow c$
  return $z^* \leftarrow low$, $y^* \leftarrow \bar{y}$, $\forall k \in S \quad x_k^* \leftarrow$ dual vars from $Solve(Slave_k)$
end

3.3 Iterative Process

The actual expected second-stage cost, $\sum_{k=1}^{K} p_k Q_k(\overline{y})$, is then evaluated by solving the second-stage problem for each scenario. Additional terms, in the form of $\{\widehat{\lambda}_k(h_k - T_k y)\}$, are added to the partial master problem to complete the iteration. Each additional term is actually another cut added to the model.

At each iteration of Benders' algorithm, then, the slave problem solution provides an upper bound for z, and, the partial master solution provides a lower bound for z. It can be proved that the above iterative procedure terminates in a finite number of iterations. An attractive feature of this algorithm is the availability of upper and lower bounds on the optimal objective value, which both converge to this value as optimality is achieved. The upper bound is generated by a sequence of feasible solutions to the problem, so the best of these may be taken as a solution if the procedure is terminated short of optimality.

The complete BD algorithm for the *SCPwLR* is presented in Algorithm 1.

4 An Illustrative Example: News Vendor Problem

We now illustrate the *SCPwLR* solution method using a modified version of the well-known "news vendor problem", a stochastic inventory replenishment problem which can be described as follows. Given a stochastic distribution for the demand of a product, what is the optimal order quantity, y^*, if only one order can be placed before actual demand is observed. Assume we have no initial inventory. The decision maker has to order an amount $y \geq 0$ at unit price $c = 3$. Unsold goods can be returned to the supplier with a salvage value of $v = 1$. In case of high demand, the firm expedites to avert an impending stockout with a cost of $e = 8$

per unit of excess demand. The maximum amount of units that can be ordered is initially $R = 20$. However, the quota can be increased by $r = [11, 13, 15, 17]$ in a nested manner, following a fixed payment of $h = [10, 12, 14, 16]$ which is also nested. Therefore, the maximum quota of 76 units can be obtained with a cost of 52. Demand is a discrete random variable denoted by ξ. ξ takes the values of $\{15(0.1), 25(0.2), 35(0.3), 45(0.3), 55(0.1)\}$, in which the values in parentheses are the probabilities, p.

A stochastic constraint program for the above problem is as follows:

$$\begin{aligned}
&\min cy + \textstyle\sum_{i=1}^{4} h_i k_i + \sum_{s=1}^{5} p_s(ex_{1s} - vx_{2s}) \\
&\text{s.t.} \\
&y + x_{1s} - x_{2s} = \xi_s, \quad s = 1, ..., 5 \\
&\textstyle\sum_{i=1}^{4} r_i k_i + 20 >= y \\
&k_i = 0 \Rightarrow k_{i+1} = 0, \quad k_i \in \{0, 1\}, \quad i = 1, ..., 3
\end{aligned}$$

where x_{1s} and x_{2s} denote expedited order and salvage amounts, respectively.

In the above model, y and k_i denote the first-stage decision variables, x_{1s} and x_{2s} are the second-stage decision variables, where s denotes a scenario, and the cost term $\sum_{s=1}^{5} p_s(ex_{1s} - vx_{2s})$ corresponds to the expected recourse cost. This problem has linear recourse. This partitioning of decision variables yields the following master and slave problems:

Master Problem
$\min \quad cy + \sum_{i=1}^{4} h_i k_i +$
$\sum_{s=1}^{5} p_s \max\{\widehat{\lambda}_s(\xi_s - y)\}$
subject to
$\sum_{i=1}^{4} r_i k_i + 20 >= y$
$k_i = 0 \Rightarrow k_{i+1} = 0 \quad i = 1, ..., 3$
$k_i \in \{0, 1\}$

Slave Problems $\quad s = 1, ..., 5$
$\max \ (\xi_s - \overline{y})\lambda_s$
subject to
$\lambda_s \leq e$
$\lambda_s \geq v$
$\lambda_s$free

Table 1 presents the step-by-step application of Benders' algorithm to our *SCPwLR* problem.

Problem	Iteration	y^*	k^*	λ^*	obj value	Lower Bound	Upper Bound
Initial values	0	0	[1,1,1,1]	–	–	$-\infty$	$+\infty$
Slave	1	–	–	[8,8,8,8,8]	340	–	340
Master	1	76	[1,1,1,1]	–	-40	-40	–
Slave	2	–	–	[1,1,1,1,1]	240	–	240
Master	2	44	[1,1,0,0]	–	155.8	155.8	–
Slave	3	–	–	[1,1,1,8,8]	155.8	–	155.8

Table 1. Steps of Benders' Decomposition

We start with a feasible solution of $y = 0$ and $k = [1, 1, 1, 1]$, which is actually the worst possible ordering policy. At the first iteration 5 independent

trivial slave problems are solved. The optimal solutions are $\lambda_{1,...,5} = 8$. Next we calculate the upper bound that these solutions imply. Eq.(5) provides an upper bound on the optimal solution to the original problem, which is 340 here. At the second step of the first iteration the partial master problem is solved with the added Benders cuts, $\max\{8(\xi_s - y)\}$, giving a lower bound of -40. Subsequent iterations of the algorithm can be followed from Table 1. In this instance the lower and upper bounds converge to the optimal solution of 155.8 in five steps.

5 Computational Experiments

This section presents computational results of using Benders' Decomposition in *SCPwLR* on the capacitated version of the Warehouse Location Problem (CWLP) (see [1], [6], [10], [11] and [13]), and stochastic version of the Template Design Problem [14].

5.1 Stochastic Capacitated Warehouse Location Problem

Let $I = \{1, ..., N\}$ be potential warehouse locations to supply a uniform product. A facility can be opened in any location $i \in I$. Opening a facility at location i has a non-negative fixed cost, f_i. Each open facility i can provide a limited amount C_i of commodity. Let $J = \{1, ..., M\}$ denote stores that are supplied by the open warehouses. For any pair (i, j) given, there is a unit production and transportation cost $g_{ij} \geq 0$. Each store can be supplied by exactly one warehouse. The probabilistic customer demands, ξ_j, are only known following stores' order placements to warehouses. Stores incur a fixed penalty cost for each unit they backlog, e_j, and fixed holding cost for each unit of excess inventory, h_j, they have. The goal is to determine a subset of the set of potential warehouse locations at which to operate warehouses, and an assignment of all clients to these facilities so as to minimize the expected total cost of operating the system. This problem is a generalisation of the well-known set covering problem and, therefore, an NP-hard problem in the strong sense.

A constraint model for the deterministic version of the above problem is given in [8]. This model is extended to comply with the stochastic demand assumption. The decision variables are:

- $k_i \in \{0, 1\}$ denoting whether warehouses i is in operation or not,
- $u_j \in I$ showing the supplier for store j,
- $y_{u_j,j} \geq 0$ is the amount warehouse u_j delivers to store j,
- $x^+_{j,s}$ and $x^-_{j,s}$ denote the excess inventory and shortage, respectively, at the end of the period at store j, if scenario $s \in S$ is realised.

The certainty equivalent CP model is

$$\min \sum_{j\in J} g_{u_j,j} y_{u_j,j} + \sum_{i\in I} f_i k_i + \sum_{j\in J}\sum_{s\in S} p_s (e_j x^-_{j,s} + h_j x^+_{j,s})$$

s.t.

$$k_{u_j} = 1 \qquad \forall j \in J$$

$$\sum_{j\in J} y_{u_j,j}(u_j = i) \le C_i \qquad \forall i \in I$$

$$y_{u_j,j} - \xi_{j,s} = x^+_{j,s} - x^-_{j,s} \qquad \forall j \in J, \forall s \in S$$

In the above formulation, the first stage (k_i, u_j, $y_{u_j,j}$) and second stage ($x^+_{j,s}$, $x^-_{j,s}$) decision variables are employed to partition the given model. In this case, the master and slave problems for the stochastic CWLP are defined as

Master Problem

$\min \sum_{j\in J} g_{u_j,j} y_{u_j,j} + \sum_{i\in I} f_i k_i +$
$\sum_{s\in S} \max\{p_s \sum_{j\in J} \widehat{\lambda}_{j,s}(\xi_{j,s} - y_{u_j,j})\}$
s.t.
$k_{u_j} = 1, \ \forall j \in J$
$\sum_{j\in J} y_{u_j,j}(u_j = i) \le C_i, \ \forall i \in I$

Slave Problems $\forall s \in S, \forall j \in J$

$\max \ (\xi_{j,s} - \overline{y}_{u_j,j})\lambda_{j,s}$
s.t.
$-h_j \le \lambda_{j,s} \le e_j$
$\lambda_{j,s}$ free

Although the slave problems are expressed in a linear program structure, in which $\lambda_{j,s}$ are dual decision variables, the simplicity of the resultant independent linear programs can be exploited to solve the problems to optimality without resorting to Linear Programming. The optimal solution to any sub-problem is in the form:

$$\lambda^*_{j,s} = \begin{cases} -h_j & \text{if } \ \xi_{j,s} - \overline{y}_{u_j,j} \le 0 \\ e_j & \text{if } \ \xi_{j,s} - \overline{y}_{u_j,j} \ge 0 \end{cases}, \quad \forall s \in S, \forall j \in J \tag{10}$$

The objective function of the master problem represents a "multiple-cut" approach to the Benders' decomposition. In this version, each scenario contributes to the cut generation process with a single cut. This excessive number of cuts may increase the size of the master problem to the point at which finding a solution is prohibitively long. Alternative approach is to aggregate scenario cuts into one, so as to reduce the detrimental effect of size on the solution performance. This so-called "single-cut" version is then in the form of $\Phi \ge \max\{\sum_s \sum_j p_s(\xi_{j,s} - y_{u_j,j})\widehat{\lambda}_{j,s}\}$, which leads to the objective function:

$$\min \sum_{j\in J} g_{u_j,j} y_{u_j,j} + \sum_{i\in I} f_i k_i + \Phi. \tag{11}$$

Although the single-cut approach does not provide cuts as strong as the multiple-cut approach, it is still computationally less expensive.

Experiments were conducted on 50 SCWLP instances using a 1.2 GHz computer. These instances vary mainly in numbers of warehouses (between 2 to 10),

stores (between 2 to 5) and scenarios (between 4 to 1331). The holding cost is taken as 1 throughout the experiments. Three different shortage cost values are used: 4, 6 and 8. The warehouse capacities are chosen in the range of [20,110]; fixed operating costs, [30,120]; random demands, [5,50]; transportation costs, [1,4]. We assume that the number of states random demand variables have in one instance is the same for all stores. Given n random variable states and j stores, the number of scenarios in an instance is n^j.

In the first step of the experiment, instances were modelled as a stochastic constraint program whose formulation is given above. The SCP models were solved using OPLStudio Solver6.0 with a time limit of 1 hour. The upper bounds for variables are chosen as follows: for y_j and x_j^-, maximum possible demand for store j; for x_j^+, the difference between the maximum and minimum demand values. The variable ordering heuristic employed assigns u_j, k_{u_j}, and $y_{u_j,j}$ in that order. The solution time (in seconds) and the number of nodes visited during search are given in Table 2 under the column heading "SCP".

Step two of the experiment repeated the first with an embedded linear relaxation. The linear relaxation enables solver to produce lower/upper bounds by solving an LP at each node of the CP search tree. The constraint solver also uses the LP solution to guide its search. The results of using embedded linear relaxation are given in Table 2 in the next two columns with a heading "SCP with LR". We also tried using linear relaxation with the variable ordering used in the first step. However, the results show that in this case linear relaxation does not prune the search space further, but incurs overhead.

In the next step of the experiment, Benders decomposition was applied to the test suite. The algorithm was implemented using ILOG Cplex9.0 and Solver6.0. The initial feasible solution was defined by adopting a no-order policy for all stores. However, since each store must be assigned to a warehouse, the least costly warehouse is operated and its fixed cost is incurred to be able to serve all stores. In this step, the "multiple-cut" version of BD was used. The results are given under the heading "BD with Multiple-Cuts". The results displayed in column "step", show the number of steps (solving a master or a slave problem defines a single step) BD takes before obtaining and proving the optimality of a solution. The column headed "Gap" gives the optimality gap for the first feasible solution, which is in the 5% gap. This experiment was repeated with "single-cuts". The results are listed under the heading "BD with Single-Cuts".

We experimented with two heuristics to test the utility of starting with a more informed solution. In heuristic-I, an expected value problem is designed to find an initial solution by replacing random demands with their maximum possible values. In heuristic-II, half the value of the maximum demands are taken as deterministic demand values. The deterministic CP models were solved to generate initial solutions. Results are presented in Table 2.

From Eq.(10), it is clear that there are only two possible values – either $-h_j$ or e_j – for any dual variables λ_j of the subproblems. Therefore, it is possible to built the "complete master" model, which can be solved to optimality without

				SCP		SCP with LR		BD with Multiple-Cuts				BD with Single-Cuts				Heuristic-I		Heuristic-II	
No	# W	# S	# Scen	Time	# Choice	Time	# Choice	Time	Step	Gap	Time	Step	Time	Step	Gap	Time	Step	Time	Step
1	2	2	4	0.07	361	0.22	634	0.07	5	2.7	0.07	4	0.03	5	2.7	0.03	4	0.04	8	0.02	6
2	2	2	9	0.22	1031	0.38	848	0.13	5	2.4	0.13	4	0.05	7	3.7	0.03	4	0.06	10	0.05	8
3	2	2	16	0.32	1687	0.56	1015	0.22	5	0	0.22	5	0.04	7	2.5	0.03	5	0.06	10	0.05	8
4	2	2	25	0.87	4464	0.74	1048	0.33	5	0	0.33	5	0.04	6	1.7	0.03	5	0.07	9	0.06	7
5	2	2	36	1.1	4913	1.2	1337	0.57	5	0	0.57	5	0.05	8	3.3	0.03	5	0.07	10	0.06	7
6	2	2	49	2.0	8068	2.3	1685	1.6	6	0.43	0.98	5	0.09	8	4.1	0.04	5	0.11	11	0.10	10
7	2	2	64	2.0	11190	2.5	2212	1.7	6	0.48	1.1	5	0.08	9	4.6	0.04	5	0.11	12	0.09	9
8	2	2	81	2.0	10724	3.3	2577	2.2	6	0.45	1.4	5	0.08	8	1.3	0.05	6	0.10	11	0.06	10
9	2	2	100	2.9	14669	4.3	3040	2.7	6	0.2	1.7	5	0.11	11	3.0	0.06	6	0.11	12	0.09	9
10	2	2	225	15	48301	39	15343	19	7	4.4	11	5	0.60	23	2.4	0.12	9	0.37	18	0.43	20
11	3	3	8	3.7	7598	11	36148	9.7	9	3.6	6.3	6	0.73	11	0.0	0.73	11	0.70	10	0.52	10
12	3	3	27	7.0	14798	24	37565	62	11	0.58	30	6	2.1	21	3.8	1.5	16	2.5	24	1.8	22
13	3	3	64	16	35607	52	36690	91	9	0.59	61	6	2.1	19	4.0	0.84	10	2.3	22	1.7	18
14	3	3	125	33	59942	128	42856	280	11	1.3	180	9	3.1	25	2.8	1.1	14	4.1	32	3.2	26
15	3	3	216	62	88078	250	51250	660	11	1.5	420	9	3.5	27	2.5	1.1	14	3.7	28	3.9	28
16	3	3	343	130	156102	390	54552	1000	11	1.3	640	8	3.8	28	3.9	1.1	14	3.5	28	3.1	32
17	3	3	512	450	363736	2200	250467	2600	11	0.82	1700	9	2.8	22	3.9	0.67	10	2.9	23	3.4	28
18	3	3	729	950	487978	3400	216757	–	–	0.33	3600	9	3.6	23	3.8	0.99	13	3.6	25	4.6	23
19	3	3	1000	1500	600508	–	–	–	–	–	–	–	4.8	25	3.1	1.3	12	4.4	25	4.6	21
20	3	3	1331	2300	797668	–	–	–	–	–	–	–	5.0	25	3.1	1.3	12	4.3	28	5.3	23
21	6	4	16	620	927903	800	1405123	1200	7	3.4	1200	6	13	9	3.1	13	8	4.6	3	4.2	4
22	6	4	81	2900	1040436	–	–	–	–	–	–	–	40	17	2.1	40	16	57	16	54	18
23	6	4	256	–	–	–	–	–	–	–	–	–	40	19	2.6	20	15	77	30	71	26
24	8	3	8	15	32817	29	70983	16	5	1.5	16	4	1.2	5	0.0	1.2	5	1.6	4	1.2	4
25	8	3	27	79	95022	64	81632	240	7	2.2	130	5	2.9	7	3.7	1.8	5	3.3	6	3.3	8
26	8	3	64	140	123451	110	80861	480	7	4.1	250	5	4.1	9	2.9	2.8	6	4.1	8	4.3	10
27	8	3	125	510	259604	370	152197	2600	7	0	2600	6	12	13	4.3	8.6	10	9.5	16	7.4	10
28	8	3	216	1600	713402	1300	341135	–	–	–	–	–	16	17	3.9	7.2	10	24	24	22	20
29	8	3	343	3100	1145120	1800	275315	–	–	–	–	–	14	19	4.8	5.6	11	29	26	16	16
30	8	3	512	–	–	3500	384398	–	–	–	–	–	31	24	4.4	9.9	13	32	26	39	26
31	8	4	16	610	698528	2000	3664085	850	5	0	850	5	9.0	5	0.0	9.1	5	16	4	8.1	4
32	8	4	81	–	–	–	–	–	–	–	–	–	21	6	2.9	13	5	27	5	16	5
33	8	4	256	–	–	–	–	–	–	–	–	–	110	15	4.5	58	11	200	18	140	14
34	8	5	32	–	–	–	–	–	–	–	–	–	41	5	0.0	41	5	67	3	54	4
35	8	5	243	–	–	–	–	–	–	–	–	–	480	8	3.7	160	5	530	5	460	7
36	9	3	8	13	28202	65	213637	40	6	0	40	6	2.1	9	4.1	2.1	9	1.1	4	0.65	4
37	9	3	27	60	79583	100	202190	350	8	1.7	200	7	4.1	10	4.1	2.8	9	3.1	7	2.8	7
38	9	3	64	140	114204	480	609580	960	9	4.8	440	6	8.1	17	1.1	6.4	14	9.6	20	8.7	16
39	9	3	125	340	189756	860	550976	–	–	4.8	1900	6	22	24	3.7	5.7	12	23	28	21	24
40	9	3	216	600	289801	1300	469528	–	–	–	–	–	18	24	3.4	5.1	12	22	30	19	26
41	9	4	16	710	1091206	1600	3387047	700	7	0	700	7	8.0	9	0.0	7.9	9	3.5	4	2.9	4
42	9	4	81	2700	972937	–	–	–	–	–	–	–	27	16	0.98	18	12	27	16	19	11
43	9	4	256	–	–	–	–	–	–	–	–	–	27	13	1.7	20	10	60	18	36	16
44	9	5	32	–	–	–	–	–	–	–	–	–	140	9	4.6	140	8	46	4	41	4
45	10	3	8	6.0	10025	21	51635	7.7	4	0	7.7	4	0.89	5	0.0	0.89	5	1.1	3	0.63	4
46	10	4	16	330	379947	790	1431662	240	4	0	240	4	5.1	5	0.0	5.1	5	9.4	4	4.2	4
47	10	4	81	2700	991063	–	–	–	–	–	–	–	23	15	4.7	18	12	31	16	18	12
48	10	4	256	–	–	–	–	–	–	–	–	–	44	19	5.0	32	14	65	18	47	19
49	10	5	32	–	–	–	–	–	–	–	–	–	97	7	0.0	97	7	37	4	28	4
50	10	5	243	–	–	–	–	–	–	–	–	–	310	11	4.4	310	10	170	5	220	8

Table 2. Experimental Results – Stochastic Capacitated Warehouse Location Problem

any Benders iteration, for this test problem. The complete master model is tried on these 50 instances. However, the observed computational performance is very discouraging. The solution times are on the average 100 times more than "BD with Single-Cuts" solution times, and 2 times more than "SCP".

The results in Table 2 show that monolithic SCP – either with variable ordering or linear relaxation – could not prove optimality in 10 instances under 1 hour. At termination, in only 2 instances the best-so-far feasible solutions were optimal. The results also point out another strong side of BD approach. If an optimality gap which is less than 5% is considered satisfactory, then the solution time is halved on average for the addressed test suite. It should also be noted that we obtained the optimal solutions in 8 out of 50 cases.

A multiple regression at confidence level 0.95 with a constant coefficient 0 gives us an insight to the BD algorithms' exact behavior on SCWL problem. The dependent variable is defined as the ratio of the solution times $st(.)$, $st(CP)/st(BD_{single})$ and independent variables of # warehouses, # stores, # scenarios. The adjusted R^2 is 0.83, which demonstrates that the overall regression model is meaningful. The regression coefficients are 4.76 (warehouses), -3.00 (stores) and 0.33 (scenarios). The corresponding t-statistics are 1.72, -0.49, 14.01, hence only the "scenarios" coefficient is significant. The results indicate that as the number of scenarios increases the performance of BD is more significant.

The two initial-solution heuristics give mixed results. In some instances the overhead of solving an additional CP model to obtain a better starting solution is not worth the effort. However, cheaper heuristics that exploit the unique structure of the problem addressed can be designed and the total solution time performance can be improved.

5.2 Stochastic Template Design Problem

The deterministic **template design problem** (prob002 in CSPLib) is described as follows. Given is a set of variations of a design, with a common shape and size and such that the number of required "pressings" of each variation is known. The problem is to design a set of templates, with a common capacity to which each must be filled, by assigning one or more instances of a variation to each template. A design should be chosen that minimises the total number of "runs" of the templates required to satisfy the number of pressings required for each variation. As an example, the variations might be for cartons for different flavours of cat food, such as fish or chicken, where ten thousand fish cartons and twenty thousand chicken cartons need to be printed. The problem would then be to design a set of templates by assigning a number of fish and/or chicken designs to each template such that a minimal number of runs of the templates is required to print all thirty thousand cartons. In the stochastic version of the problem, the demand for each variation is uncertain.

Proll and Smith address this problem by fixing the number of templates and minimising the total number of pressings [14]. We adopt their model herein, extending it to comply with the stochastic demand assumption. We use the following notation: N, number of variations; T, number of templates (T=2, for

all instances of the test suit); S, number of slots on each template; K, number of scenarios (demand for each variation is uncertain); c_h, scrap cost; c_p, shortage cost; $a_{i,j}$, number of slots designated to variation i, on template j; R_j, number of required "runs" of template j; x_i, an auxiliary variable; $d_{i,k}$, demand for variation i in scenario k; $e_{i,k}$, total scrap (variation i in scenario k); $b_{i,k}$, total shortage (variation i in scenario k); p_k, probability of observing scenario k.

The certainty equivalent CP model is:

$\min \ \sum_{i=1}^{N} \sum_{k=1}^{K} p_k(c_p b_{ik} + c_h e_{ik})$
subject to
$\sum_{i=1}^{N} a_{ij} = S, \forall j \in \{1, ..., T\}$,
$\sum_{j=1}^{T} a_{ij} R_j = x_i, \forall i \in \{1, ..., N\}$, and
$x_i = d_{ik} + e_{ik} - b_{ik}, \forall i \in \{1, ..., N\}, \forall k \in \{1, ..., K\}$.

In the above formulation, the first stage (a_{ij}, R_j) and second stage (x_i, e_{ik}, b_{ik}) decision variables are employed to partition the given model. In this case, the master and slave problems for the Stochastic Template Design Problem are defined as

Master Problem
$\sum_{k=1}^{K} \max\{\sum_{i=1}^{N} p_k(d_{ik} - x_i)\widehat{\lambda}_{ik}\}$
subject to
$\sum_{i=1}^{N} a_{ij} = S, \ \forall j \in \{1, ..., T\}$
$\sum_{j=1}^{T} a_{ij} R_j = x_i, \ \forall i \in \{1, ..., N\}$

Slave Problems,
$\forall i \in \{1, ..., N\}, \forall k \in \{1, ..., K\}$
$\max \ (d_{ik} - \overline{x}_i)\lambda_{ik}$
subject to
$-c_h \leq \lambda_{ik} \leq c_p$, λ_{ik} free

				SCP		Complete Benders		BD with Multiple-Cuts						BD with Single-Cuts				
No	N	S	Scen	Time	#choice	Time	#choice	Time	step	gap	Time	step		Time	step	gap	Time	step
1	3	6	10	340	846,584	190	845,660	4,400	13	0	4,400	13		9.5	33	4.1	7.9	29
2	3	9	10	1,400	3,401,506	780	3,400,897	–	–	–	–	–		8.1	31	4.8	7.4	26
3	4	2	10	230	703,957	140	694,949	3,100	27	4.0	2,100	25		62	43	3.7	42	36
4	4	3	10	470	1,153,468	270	1,145,436	5,500	23	0.6	3,100	21		41	39	4.7	26	32
5	4	4	10	1,900	4,165,262	1,100	4,159,062	–	–	–	–	–		190	49	4.3	130	40
6	4	5	10	690	1,630,615	380	1,627,743	–	–	–	–	–		380	41	3.4	380	36
7	4	6	10	–	–	5,700	20,909,785	–	–	–	–	–		670	52	3.5	510	44
8	4	7	10	–	–	–	–	–	–	–	–	–		820	53	4.8	600	40
9	4	8	10	–	–	–	–	–	–	–	–	–		1,800	50	4.4	1,000	37
10	4	9	10	–	–	–	–	–	–	–	–	–		5,900	49	3.3	4,900	43
11	4	6	11	–	–	6,800	21,302,918	–	–	–	–	–		480	51	4.1	410	44
12	4	6	12	–	–	–	–	–	–	–	–	–		1,100	52	4.6	640	41
13	4	6	13	–	–	–	–	–	–	–	–	–		190	58	4.9	160	43
14	4	6	14	–	–	–	–	–	–	–	–	–		1,300	57	3.3	760	42
15	4	6	15	–	–	–	–	–	–	–	–	–		790	53	3.5	780	42
16	4	6	16	–	–	–	–	–	–	–	–	–		1,800	55	4.4	1,500	40
17	4	6	17	–	–	–	–	–	–	–	–	–		540	54	4.8	370	38
18	4	6	18	–	–	–	–	–	–	–	–	–		290	53	3.7	230	40
19	4	6	19	–	–	–	–	–	–	–	–	–		590	61	3.9	500	49
20	4	6	20	–	–	–	–	–	–	–	–	–		400	59	4.2	360	40

Table 3. Experimental Results – Stochastic Template Design Problem

Experiments, summarised in Table 3, were conducted on 20 stochastic template design instances using an Intel Centrino 2GHz computer with 1GB RAM. Allowed solution time was 2 hours. A dash in the table indicates this time was exceeded without finding the optimal solution.

We compared four different solution methods: stochastic constraint programming (solved using Ilog OPLStudio and Solver 6.0), and three versions of Benders Decomposition (solved using Ilog Cplex 9.0 and Solver 6.0). In the first, the complete master problem, which can be solved to optimality without any Benders iterations, was built and solved using Solver 6.0. In the second and third, the multiple cut and single cut versions of BD were used. For the stochastic constraint program and complete Benders formulation, the number of choice points explored to find and prove optimality are shown. For the latter two Benders variations, the "step" column indicates the number of steps (solving a master or a slave problem is counted as a step) BD takes before proving optimality. The "Gap" column gives the optimality gap for the first feasible solution found with an optimality gap of 5% or less.

The results show that monolithic SCP can solve only the smaller instances in the time allowed. The performance of the complete Benders formulation is comparable, giving better times and solving two more instances. The performance of the two BD approaches is at polar opposites. The multiple-cut variant performs relatively poorly, solving fewer instances than SCP and taking much more time. However, the single-cut variant is clearly the strongest approach that we tested, solving all 20 instances.

6 Conclusion

This paper has aimed to enhance the effectiveness of the stochastic constraint programming framework by extending the use of the well-known Benders' decomposition algorithm, which has proved to be useful in mathematical programming, to solve stochastic constraint programs with linear recourse, *SCPwLR*. First and second stage decision variables are used to decompose the stochastic constraint program into master and slave problems. The unique structure of stochastic constraint programs, which is based upon a scenario tree representation, yields independent slave sub-problems, one for each scenario considered in the scenario tree. This natural slave problem decomposition has the obvious benefit of solving a set of small problems, and hence to a degree relieving the difficulty of dealing with a large scenario tree.

In our test problems, it was shown that the slave problem decomposes into trivial scenario problems, each of which is a boundary value problem and can be solved simply by checking the objective function coefficient and deciding whether the single decision variable takes the lower or the upper limit of its domain. Computational experiments confirmed the potential of Benders' decomposition method as an efficient solution algorithm for these problems.

An attractive feature of this algorithm is the availability of upper bounds and lower bounds produced by the slave and master problem solutions, respectively. The upper bound is generated by a sequence of feasible solutions to the problem, so the best of these may be taken as a solution if the procedure is terminated short of optimality. Furthermore, the best-so-far lower bound can be used to produce a metric for the optimality gap.

Although at present we consider only 2-stage *SCPwLR*, the ideas presented here can be generalised to n-stage *SCPwLR* without loss of generality. In this case, Benders' algorithm is applied in a recursive manner. We expect the performance discrepancy between the monolithic model and the decomposed model to be magnified as the number of stages grows. A further extension of this work should consider the case in which the linear recourse assumption is relaxed and recourse actions with complex structures are allowed. Hooker and Ottosson [7] provide a method for generating Benders cuts in such situations by generalising the linear programming dual of a sub-problem to an "inference" dual. We will adopt this approach for general stochastic CSP.

Acknowledgements: S. Armagan Tarim is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 00/PI.1/C075. Ian Miguel is supported by a UK-Royal Academy of Engineering/EPSRC Research Fellowship.

References

1. Beasley, J.E.: An algorithm for solving large capacitated warehouse location problems. European Journal of Operational Research, **33** (1988) 314–325
2. Benders, J. F.: Partitioning Procedures for Solving Mixed-Variables Programming Problems. Numerische Mathematik, **4** (1962) 238–252
3. Benoist, T., Gaudin, E., Rottembourg, B.: Constraint Programming Contribution to Benders Decomposition: A Case Study. Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, LNCS **2470** (2002) 603–617
4. Eremin, A., Wallace, M.: Hybrid Benders Decomposition Algorithms in Constraint Logic Programming. Principles and Practice of Constraint Programming, Proceedings of the 7th International Conference, CP 2001, Paphos, Cyprus, LNCS **2239** (2001) 1–15
5. Geoffrion, A. M.: Generalized Benders decomposition. Journal of Optimization Theory and Applications, **10** (1972) 237-260
6. Holmberg, K., Ronnqvist, M., Yuan, D.: An exact algorithm for the capacitated facility location problem with single sourcing. European Journal of Operational Research, **113** (1999) 544–559
7. Hooker, J. N., Ottosson, G.: Logic-based Benders decomposition. Mathematical Programming, **96** (2003) 33–60
8. ILOG Inc.: OPL Studio 3.7, Studio Users Manual, (2003)
9. Jain, V., Grossmann, I. E.: Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. INFORMS Journal on Computing, **13** (2001) 258–276
10. Khumawala, B. M.: An Efficient Branch-Bound Algorithm for the Warehouse Location Problem. Management Science. **18** (1972) 718–731
11. Krarup, J., Pruzan, P. M.: The simple plant location problem: Survey and synthesis. European Journal of Operational Research. **12** (1983) 36–81
12. Manandhar, S., Tarim, S. A., Walsh. T.: Scenario-Based Stochastic Constraint Programming. Proceedings of IJCAI-2003, Acapulco, Mexico. (2003) 257–262
13. Mirchandani, P. B., Francis, R. L.: Discrete Location Theory. John Wiley and Sons, (1990)
14. Proll, L., Smith, B.M. Integer linear programming and constraint programming approaches to a template design problem. INFORMS Journal on Computing 10(3):265–275, 1998.
15. Tarim, S. A., Manandhar, S., Walsh. T.: Stochastic Constraint Programming. submitted for publication, (2004)
16. Thorsteinsson, E. S.: Branch-and-Check: A Hybrid Framework Integrating Mixed Integer Programming and Constraint Logic Programming. Principles and Practice of Constraint Programming, Proceedings of the 7th International Conference, CP 2001, LNCS **2239** (2001) 16–30
17. Xia, Q., Eremin, A., Wallace. M.: Problem Decomposition for Traffic Diversions. Proceedings of the 1st International Conference, Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2004, LNCS **3011** (2004) 348–363