LA-UR-10-05696

Approved for public release;
distribution is unlimited.

Title:

Spacial and Objective Decompositions for Very Large SCAPs

Author(s):

Carleton Coffrin
Pascal Van Hentenryck
Russell Bent

Intended for:

2011 Conference on Integration of AI, CP and OR.

Spatial and Objective Decompositions for Very Large SCAPs

Carleton Coffrin[1], Pascal Van Hentenryck[1], and Russell Bent[2]

[1] Brown University, Providence RI 02912, USA
[2] Los Alamos National Laboratory, Los Alamos NM 87545, USA

Abstract. This paper reconsiders the single commodity allocation problem (SCAP) for disaster recovery, which determines where and how to stockpile a commodity before a disaster and how to route the commodity once the disaster has hit. It shows how to scale the SCAP algorithm proposed in [1] to a geographical area with up to 1,000 storage locations (over a million decision variables). More precisely, the paper shows that spatial and objective decompositions are instrumental in solving SCAP problems at the state scale (e.g., for the state of Florida). The practical benefits of these decompositions are demonstrated on large-scale hurricane disaster scenarios generated by Los Alamos National Laboratory using state-of-the-art disaster simulation tools.

1 Background and Motivation

Every year, considerable human and monetary resources are spent to prepare for, and recover from, seasonal hurricanes. Existing procedures rely on the experience of policy makers but they are often ad-hoc and do not exploit recent progress in optimization to address natural disasters more effectively. Our earlier research [1] demonstrated the benefits of optimization technology to meet the population needs and to reduce storage and transportation costs by using a two-stage stochastic optimization problems with explicit scenarios generated by the National Hurricane Center (NHC) of the National Weather Service in the United States. However, only disasters with up to 100 storage locations (city scale) were considered, although large-scale planning may require as many as 1,000 storage locations (state scale). Indeed, the start-of-the-art algorithm in [1], and its underlying MIP model, have difficulties scaling to problems with 250 storage locations and runs out of memory on larger instances.

This paper shows how to scale the approach for disasters at the state scale using spatial and objective decompositions. The spatial decomposition performs a geographic clustering of the repositories and aggregates the flows across the clusters, thus reducing the number of decision variables considerably. The objective decomposition applies when the SCAP objective function is lexicographic: It separates the decisions taken for meeting the demands and reducing travel time. Experimental results demonstrate the benefits of both approaches on large and very large instances respectively. Given the sizes and complexity of the models considered here, our results are purely empirical: Their practicability is demonstrated by showing improvements on the practice in the field. Note also that the resulting approaches are now deployed and are activated each time a hurricane of category 3 or above threatens the coast of the United States.

The rest of the paper is organized as follows. Section 2 of this paper reviews related work. Section 3 presents a mathematical formulation of the SCAP and Section 4 reviews the approach presented in [1]. Sections 5 and 6 present the novel decomposition techniques. Section 7 reports the experimental results and Section 8 concludes the paper.

2 Previous Work

Humanitarian logistics has been investigated since the 1990s but has received increased attention in recent years due to the increase in major disasters [2–5]. Humanitarian logistics gives rise optimization problems combining aspects of inventory routing, supply chain management, warehouse location, and vehicle routing, creating novel challenges for existing technology [2, 3]. They often combine some, or all, of the following features:

1. **Multi-Objective Functions** - High-stake disaster situations often have to balance conflicting objective goals (e.g. operational costs, speed of service, and unserved customers) [6–9, 1].
2. **Non-Standard Objective Functions** - A makespan time objective in VRPs [6, 10, 1] or equitability objectives [8].
3. **Arbitrary Side Constraints** - Limited resources, a fixed vehicle fleet [8, 1], fixed latest delivery time [6, 8], or a insufficient budget [7, 11, 1].
4. **Stochastic Aspects** - Disasters are inherently unpredictable. Preparations and recovery plans must be robust with respect to many scenarios [7, 9, 1].

Applications in humanitarian logistics are studied at a variety of scales in space and time. Some problems consider a global scale with time measured in days and weeks [7], while others focus on the minute-by-minute details of delivering supplies from local warehouses directly to the survivors [6, 8, 1, 12]. This paper considers the so-called “last mile” of distribution which involves warehouse selection and customer delivery at the city and state scales.

Humanitarian logistics applications have been mostly formulated as mixed integer programming (MIP) models, which often do not scale to real-world instances [8, 6, 12]. Moreover, MIP solvers have been shown to have severe difficulties with some of their unique features even when problem sizes are small (e.g., minimizing the latest delivery time in VRPs [10]). Our earlier research [1] demonstrated that hybrid optimization and decomposition methods can yield high-quality solutions to such challenges and scale to real-world instances. This work extends those results and shows that spatial and objective decompositions provide significant scaleability. To the best of our knowledge, this is the first time that SCAPs with over 100 storage locations have been solved.

3 The Single Commodity Allocation Problem (SCAP)

In formalizing SCAPs, a populated area is represented as a graph $G = \langle N, E\rangle$ where N represents the locations of interest to the allocation problem: Sites

Given:
- Repositories: $i \in R$
  - Capacity: RC_i
  - Investment Cost: RI_i
  - Maintenance Cost: RM_i
- Vehicles: $i \in V$
  - Capacity: VC
  - Start Depot: H_i^+
  - End Depot: H_i^-
- Scenario Data: $i \in S$
  - Scenario Probability: P_i
  - Available Sites: $AR_i \subset R$
  - Site Demand: $D_{i,j \in R}$
  - Travel Time Matrix: $T_{i,1..l,1..l}$
- Weights: W_x, W_y, W_z
- Budget: B

Output:
- The amount stored at each warehouse
- Delivery schedules for each vehicle

Minimize:
- $W_x *$ Unserved Demands $+$
- $W_y * MAX_{i \in V}$ Tour Time$_i +$
- $W_z *$ Investment Cost $+$
- $W_z *$ Maintenance Cost

Subject To:
- Vehicle and site capacities
- Vehicles start and end locations
- Costs $\leq B$

Notes:
- Every warehouse that stores commodities must be visited at least once

Fig. 1. The Single Commodity Allocation Problem Specification.

requiring the commodity after the disaster (e.g., hospitals, shelters, and public buildings) and vehicle storage depots. The required commodity can be stored at any node of the graph subject to some side constraints and the graph edges, E, have weights representing travel times. The weights on the edges form a metric space but it is not Euclidean due to the transportation infrastructure. Moreover, travel times can vary in different disaster scenarios due to road damage. The primary outputs of a SCAP are (1) the amount of commodity to be stored at each node; (2) for each scenario and each vehicle, the best plan to deliver the commodities. Figure 1 summarizes the entire problem, which we now describe in detail.

Objectives The objective function aims at minimizing three factors: (1) The amount of unsatisfied demands; (2) the time it takes to meet those demands; (3) the cost of storing the commodity. Since these values are not expressed in the same units, it is not always clear how to combine them into a single objective function. Furthermore, their relative importance is typically decided by policy makers on a case-by-case basis using weights W_x, W_y, and W_z. Note that the routing objective is to minimize the time of the last delivery, which is required by the Department of Homeland Security in the United States. Minimizing the time of the last delivery is a very difficult aspect of this problem as demonstrated in [10]. However, when solved with a combination of large neighborhood search and constraint programming, the stochastic storage decisions quickly become the most difficult aspect as the number of storage locations increases.

Side Constraints Each repository $i \in R$ has a maximum capacity RC_i to store the commodity. It also has a one-time initial cost RI_i (the investment cost) and an incremental cost RM_i for each unit of commodity to be stored. As policy makers often work within budget constraints, the sum of all costs in the system must be less than a budget B. Every repository can act as a warehouse and a customer and its role changes on a scenario-by-scenario basis depending on site

```
MULTI-STAGE-SCAP($SCAP\ \mathcal{G}$)
1  $\mathcal{D} \leftarrow StochasticStorageProblem(\mathcal{G})$
2  for $s \in S$
3  do $\mathcal{C} \leftarrow CustomerAllocationProblem(\mathcal{G}_s, \mathcal{D}_s)$
4      for $w \in R$
5      do $\mathcal{T} \leftarrow RepositoryPathRoutingProblem(\mathcal{G}_s, \mathcal{C}_w)$
6      $\mathcal{I} \leftarrow AggregateFleetRoutingProblem(\mathcal{G}_s, \mathcal{T})$
7      $\mathcal{F}_s \leftarrow PathBasedFleetRoutingProblem(\mathcal{G}_s, \mathcal{T}, \mathcal{I})$
8  return $\mathcal{F}$
```

Fig. 2. The Hybrid Stochastic Optimization Algorithm for Solving SCAPs.

availability and demands. Additionally, if a repository is acting as a warehouse for its own demands a vehicle must still visit that location before the stored commodities are available for consumption.

SCAPs also feature a fleet of V vehicles which are homogeneous in terms of their capacity VC. Each vehicle $i \in V$ has a unique starting depot H_i^+ and ending depot H_i^-. Unlike classic vehicle routing problems [13], customer demands in SCAPs often exceed the vehicle capacity and hence multiple deliveries are often required to serve a single customer.

Stochasticity SCAPs are specified by a set of S different disaster scenarios. Scenario $i \in S$ has an associated probability P_i and specifies the set AR_i of sites which remain intact after the disaster. Moreover, scenario i specifies, for each repository $j \in R$, the demand D_{ij} and site-to-site travel times $T_{i,1..l,1..l}$ (where $l = |N|$) which capture the damages to the transportation infrastructure.

4 The Basic Approach

This section reviews the state-of-the-art algorithm for solving the SCAP problem [1]. This multi-stage algorithm, depicted in Figure 2, decomposes the storage, customer allocation, and routing decisions. The stages and the key decisions of each stage are as follows: (1) *Stochastic Storage*: Which repositories store the commodity and how much do they store? (2) *Customer allocation*: How is the stored commodity allocated to each customer? (3) *Repository routing*: For each repository, what is the best customer distribution plan? (4) *Fleet routing*: How to visit the repositories to minimize the time of the last delivery? The decisions of each stage are considered independently and use the optimization technique most appropriate to their nature. The first two stages are formulated as MIPs, the third stage is solved optimally using constraint programming (CP), and the fourth stage uses large neighborhood search (LNS) and CP.

This work only considers modifications to the Stochastic Storage Model (SSM) and uses identical algorithms for the customer allocation and routing aspects of the problem. Hence, we only review the SSM in detail. The SSM captures the cost and demand objectives precisely but approximates the routing aspects. In particular, the SSM only considers the time to move the commodity from the repository to a customer, not the maximum delivery time. Let D be a

Variables:

$Stored_i \in (0, RC_i)$ - Units stored at repository i
$Open_i \in \{0, 1\}$ - Non-zero storage at repository i

Second stage variables for each scenario s:
$Outgoing_{si} \in (0, RC_i)$ - Total units shipped from repository i
$Incoming_{si} \in (0, D_{si})$ - Total units coming to repository i
$Unsatisfied_{si} \in (0, D_{si})$ - Demand not satisfied at repository i
$Sent_{sij} \in (0, RC_i)$ - Units shipped from repository i to repository j

Minimize:

$$W_x \sum_{s \in S} P_s \sum_{i \in R} Unsatisfied_{si} + W_z \sum_{i \in R} (RI_i \; Open_i + RM_i \; Stored_i) +$$
$$W_y \sum_{s \in S} P_s \sum_{i \in R} \sum_{j \in R} T_{sij} \; Sent_{sij} / VC$$

Subject To:

$$\sum_{i \in R} (RI_i \; Open_i + RM_i \; Stored_i) \leq B \qquad (1)$$
$$RC_i \; Open_i \geq Stored_i \qquad \forall i \in R \qquad (2)$$
$$Incoming_{si} + Unsatisfied_{si} = D_{si} \qquad \forall s \in S, i \in R \qquad (3)$$
$$Outgoing_{si} \leq Stored_i \qquad \forall s \in S, i \in R \qquad (4)$$
$$\sum_{j \in R} Sent_{sij} = Outgoing_{si} \qquad \forall s \in S, i \in R \qquad (5)$$
$$\sum_{j \in R} Sent_{sji} = Incoming_{si} \qquad \forall s \in S, i \in R \qquad (6)$$
$$Outgoing_{si} = 0 \qquad \forall s \in S, i \notin AR_s \qquad (7)$$

Fig. 3. The MIP Formulation for the Stochastic Storage Model (SSM).

set of delivery triples of the form $\langle source, destination, quantity \rangle$. The delivery-time component of the objective is replaced by

$$W_y \sum_{\langle s,d,q \rangle \in D} T_{sd} \frac{q}{VC}$$

Figure 3 presents the SSM formulation which scales well with the number of disaster scenarios since the number of integer variables only depends on the number of repositories. The meaning of the decision variables is explained in the figure. The objective function sums the unsatisfied demands for each scenario, the investment and maintenance costs, and the shipping costs for each scenario. The second stage costs are obviously multiplied by the scenario probabilities. Constraint (1) captures the budget constraint and constraint (2) ensures that a repository is open if it stores the commodity. Constraint (3) states for each scenario that the unsatisfied demand of a repository is the repository's demand minus the incoming supply (which can include local storage). Constraint (4) expresses that the supply shipped from repository i cannot exceed the amount of commodity stored at repository i. Constraints (5–6) connect the sent, incoming, and outgoing variables and constraint (7) ensures that damaged repositories ship no commodity.

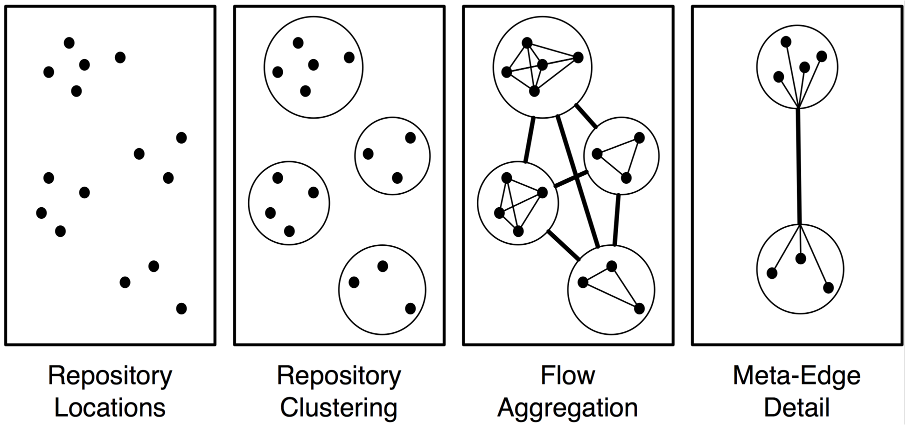

Fig. 4. Storage Clustering and Flow Aggregation.

The experimental results in [1] indicate that the fleet-routing stage of the algorithm is the dominant factor in the algorithm runtime. However, for instances with more than 100 storage locations, the SSM quickly dominates the runtime (see Section 7 for numerical evidence). The next two sections present two alternative models for the stochastic storage problem that provide significant benefits for scalability. Both stochastic storage models rely on a key observation: In the baseline algorithm (Figure 2), a customer allocation is computed in the SSM and then recomputed in the customer allocation stage (once the uncertainty is revealed). This means, when a customer allocation stage is used, only the storage decisions are a necessary output of the SSM. The two new SSM models achieve better performance by approximating or ignoring the customer allocation in the first-stage storage problem.

5 Spatial Decomposition

In the SSM, the number of variables required for the customer allocation is quadratic in the number of repositories and multiplicative in the number of scenarios (i.e., $|S||R|^2$). The number of variables can easily be over one million when the number of repositories exceeds two hundred. Problems of this size can take up to 30 seconds to solve with a linear-programming solver and the resulting MIP can take several hours to complete. Our goal is thus to reduce the number of variables in the MIP solver significantly, without degrading the quality of the solutions too much.

The Aggregate Stochastic Storage Model (ASSM) is inspired by the structure of the solutions to the baseline algorithm. Customers are generally served by storage locations that are *nearby* and commodities are only transported over large distances in extreme circumstances. We exploit this observation by using a geographic clustering of the repositories. The clustering partitions the set of repositories R into C clusters and the repositories of a cluster $i \in C$ are denoted by CL_i. For a given clustering, we say that two repositories are *nearby* if they are

Calculate:

$$CS_c = \sum_{i \in CL_c} RC_i \qquad \text{- Total storage in cluster } c$$

$$CD_{sc} = \sum_{i \in CL_c} D_{si} \qquad \text{- Total demand in cluster } c \text{ in scenario } s$$

Variables:

$Stored_i \in (0, RC_i)$ - Units stored at repository i

$Open_i \in \{0, 1\}$ - Non-zero storage at repository i

Second stage variables for each scenario s:

$Unsatisfied_{si} \in (0, D_{si})$ - Unsatisfied demands at repository i

$Incoming_{sic} \in (0, D_{si})$ - Units shipped from cluster c to repository i

$Outgoing_{sic} \in (0, RC_i)$ - Units shipped from repository i to cluster c

$Sent_{sij} \in (0, min(RC_i, D_{sj}))$ - Units shipped from repository i to repository j

$Link_{scd} \in (0, min(CS_c, CD_{sd}))$ - Units sent from cluster c to cluster d

Minimize:

$$W_x \sum_{s \in S} P_s \sum_{i \in R} Unsatisfied_{si} + W_z \sum_{i \in R} (RI_i \ Open_i + RM_i \ Stored_i) +$$
$$W_y \sum_{s \in S} P_s \sum_{c \in C} \sum_{i \in CL_c} \sum_{j \in CL_c} T_{sij} \ Sent_{sij}/VC + W_y \sum_{s \in S} P_s \sum_{c \in C} \sum_{d \in C} CT_{scd} \ Link_{scd}/VC$$

Subject To:

$$\sum_{i \in R} (RI_i \ Open_i + RM_i \ Stored_i) \le B \tag{1}$$

$$RC_i \ Open_i \ge Stored_i \qquad \forall i \in R \tag{2}$$

$$\sum_{j \in R} Sent_{sji} + \sum_{c \in C} Incoming_{sic} + Unsatisfied_{si} = D_{si} \qquad \forall s \in S, i \in R \tag{3}$$

$$\sum_{j \in R} Sent_{sij} + \sum_{c \in C} Outgoing_{sic} \le Stored_i \qquad \forall s \in S, i \in R \tag{4}$$

$$\sum_{i \in CL_c} Outgoing_{sid} = Link_{scd} \qquad \forall s \in S, c \in C, d \in C \tag{5}$$

$$\sum_{i \in CL_d} Incoming_{sic} = Link_{scd} \qquad \forall s \in S, c \in C, d \in C \tag{6}$$

$$Sent_{sij} = 0 \qquad \forall s \in S, i \notin AR_s, j \in R \tag{7}$$

$$Outgoing_{sic} = 0 \qquad \forall s \in S, i \notin AR_s, c \in C \tag{8}$$

Fig. 5. The MIP Formulation for the Aggregate Stochastic Storage Model (ASSM).

in the same cluster; otherwise the repositories are *far away*. Nearby repositories have a tightly-coupled supply and demand relationship and hence the model needs as much flexibility as possible in mapping the supplies to the demands. This flexibility is achieved by allowing commodities to flow between each pair of repositories within a cluster (as was done in SSM). When repositories are far away, the precise supply and demand relationship is not as crucial since the warehouse to customer relationship is calculated in the customer allocation stage of the algorithm. As a result, it is sufficient to reason about the aggregate flow moving between two clusters at this stage of the algorithm. The aggregate flows are modeled by introducing *meta-edges* between each pair of clusters. If some demand from cluster $a \in C$ must be met by storage locations from cluster $b \in C$, then the sending repositories CL_b pool their commodities in a single *meta-edge* that flows from b to a. The receiving repositories CL_a then divide up the pooled commodities in the *meta-edge* from b to meet all of their demands. Additionally, if each *meta-edge* is assigned a travel cost, the *meta-edge* can approximate the number of trips required between two clusters by simply dividing the total amount of commodities by the vehicle capacity, as is the case for all the other flow edges. Figure 4 visually indicates how to generate the flow decision variables for the clustered problem and how commodities can flow on *meta-edges* between customers in different clusters.

As stated above, the number of variables in the SSM is quadratic in the number of repositories. Given a clustering $CL_{i \in C}$, the number of variables in the clustered storage model is (1) quadratic within each cluster (i.e., $\sum_{i \in C} |CL_i|^2$); (2) quadratic in the number of clusters, (i.e., $|C|^2$); (3) and linear in the repositories connections to the clusters (i.e., $2|R||C|$). The exact number of variables clearly depends on the considered clustering. However, given a specific number $|C|$ of clusters, a lower bound on the number of variables is obtained by dividing the repositories evenly among all the clusters, and the best possible variable reduction on a problem of size n with c clusters and s scenarios is $s\ (\frac{n^2}{c} + 2nc + c^2)$.

Given a clustering $CL_{i \in C}$ and cluster to cluster travel times CT_{scc} for each scenario, the ASSM is presented in Figure 5. The meaning of the decision variables is explained in the figure. The objective function has two terms for the delivery times, one for the shipping between repositories inside a cluster and one for shipping between clusters. Constraints (1–2) are the same as in the SSM model. Constraints (3–4) take into account the fact that the commodity can be shipped from repositories inside the clusters and from clusters. Constraints (5–6) aggregate the outgoing and incoming flow for a cluster, while constraints (7–8) express the damage constraints. Note that the array of variables $Sent_{sij}$ is sparse and only includes variables for repositories inside the same cluster (this is not reflected in the notations for simplicity).

6 Objective Decomposition

The ASSM significantly decreases the number of variables but it still requires creating a quadratic number of variables for each cluster. Since this is multiplied by the number of scenarios, the resulting number of variables can still be prohibitive for very large instances. This section presents an objective decomposition which applies when the objective is lexicographic, i.e., when policy makers set the values of the weights such that $W_x \gg W_y \gg W_z$, which is often the case in practice. Let us contemplate what this means for the behavior of the model algorithm as the budget parameter B is varied. With a lexicographic objective, the model will first try to meet as many demands as possible. If the demands can be met, it will reduce delivery times until it cannot be reduced further or the budget is exhausted. As a result, the optimization with a lexicographic objective exhibits three phases as B increases. In the first phase, the satisfied demands, routing times, and costs increase steadily. In the second phase, the satisfied demands remain at a maximum, the routing times decrease, and the costs increase. In the last phase, the satisfied demands remain at a maximum, the routing times

Calculate:

$SD_s = \sum_{i \in R} D_{si}$ - Total demand in scenario s

Variables:

$Stored_i \in (0, RC_i)$ - Units stored at repository i

$Open_i \in \{0, 1\}$ - Non-zero storage at repository i

$Used_s \in (0, SD_s)$ - Units used in scenario s

Minimize:

$$\sum_{s \in S} P_s \, (SD_s - Used_s)$$

Subject To:

$$RC_i \; Open_i \geq Stored_i \qquad \forall i \in R \quad (1)$$

$$\sum_{i \in AR_s} Stored_i \geq Used_s \qquad \forall s \in S \quad (2)$$

$$\sum_{i \in R} (RI_i \; Open_i + RM_i \; Stored_i) \leq B \quad (3)$$

Fig. 6. Phase 1 of the Lexicographic Stochastic Storage Model (LSSM-1).

remain at a minimum, and the costs plateau even when B increases further. The experimental results from [1] confirm this behavior.

The Lexicographic Stochastic Storage Model (LSSM) assumes that the objective is lexicographic and solves the first phase with a much simpler (and faster) model. The goal of this phase is to use the available budget in order to meet the demands as best possible and it is solved with a two-stage stochastic allocation model that ignores the customer allocation and delivery time decisions. Since each scenario s has a total demand SD_s that must be met, it is sufficient to maximize the expected amount of demands that can be met, conditioned on the stochastic destruction of storage locations. Figure 6 presents such a model. The meaning of the decision variables is explained in the figure.

During the first phase, the model in Figure 6 behaves similarly to the SSM for a lexicographic objective. But the model does not address the delivery times at all, since this would create a prohibitive number of variables. To compensate for this limitation, we use a second phase whose idea can be summarized by the following greedy heuristic: "if all the demands can be met, use the remaining budget to store as much additional commodity as possible to reduce delivery times". This greedy heuristic is encapsulated in another MIP model (LSSM-2) presented in Figure 7. LSSM-2 utilizes the remaining budget while enforcing the decisions of the first step by setting the lower bound of the $StoredEx_i$ variables to the value of the $Stored_i$ variables computed by LSSM-1. This approximation is rather crude but produces good results on actual instances (see Figures 9 and 10 in Section 7). Our future work will investigate how to improve this formulation by taking account of customer locations, while still ignoring travel distances.

The resulting approach is less flexible than the SSM and ASSM approaches because it ignores the weighting factors W_x, W_y, and W_z. However, it produces a significant increase in performance by decreasing the number of decision variables from quadratic to linear. The asymptotic reduction is essential for scaling the algorithm to very large instances. Note that it is well-known in the

Variables:

$StoredEx_i \in (Stored_i, RC_i)$ - Units stored at repository i

$OpenEx_i \in \{0, 1\}$ - Non-zero storage at repository i

Maximize:

$$\sum_{i \in R} StoredEx_i$$

Subject To:

$$RC_i \; OpenEx_i \geq StoredEx_i \qquad \forall i \in R \qquad (1)$$

$$\sum_{i \in R} (RI_i \; OpenEx_i + RM_i \; StoredEx_i) \leq B \qquad (2)$$

Fig. 7. Phase 2 of the Lexicographic Stochastic Storage Model (LSSM-2).

goal-programming community that lexicographic multi-objective programs can be solved by a series of single-objective problems [14]. The sub-objectives are considered in descending importance and, at each step, one sub-objective is optimized in isolation and side constraints are added to enforce the optimization of the previous steps. Our decomposed storage model follows the same schema, except that the second step is necessarily approximated due to its size.

7 Benchmarks and Results

Benchmarks The benchmarks were produced by Los Alamos National Laboratory and are based on the infrastructure of the United States. The disaster scenarios were generated by state-of-the-art hurricane simulation tools similar to those used by the National Hurricane Center [15]. The problem sizes and algorithm parameters are presented in Table 1. The *Trip Lower Bounds* are simply the total amount of commodities that are shipped divided by the vehicle capacity. These values are included because they are a good metric for the routing difficulty of a benchmark. The amount of commodities that need to be moved can vary significantly from scenario to scenario. Therefore, we present both the smallest and the largest trip bounds across all the scenarios. Benchmarks 3 and 6 feature scenarios where the hurricane misses the region; this results in the minimum trip bound being zero. This is important since any algorithm must be robust with respect to empty disaster scenarios which arise in practice when hurricanes turn away from shore or weaken prior to landfall. The algorithm parameters include a runtime cap for the client allocation and fleet routing subproblems (defined in [1]), and the number of clusters that will be used in the ASSM model. All of the experimental results have fixed values of W_x, W_y, and W_z satisfying the field constraint $W_x \gg W_y \gg W_z$ and we vary the value of the budget B to evaluate the algorithm (as was done in [1]). The results are consistent across multiple weight configurations, although there are variations in the problem difficulties.

The Algorithm Implementation and the Baseline Algorithm The algorithms were implemented in the COMET system [16] and the experiments were run on Intel Xeon CPU 2.80GHz machines running 64-bit Linux Debian. To validate our

Benchmark	$\|R\|$	$\|V\|$	$\|S\|$	Min Trip Lower Bound	Max Trip Lower Bound	CA Timeout	Fleet Timeout	Clusters
BM1	25	4	3	6	27	30	10	4
BM2	25	5	3	60	84	30	20	4
BM3	25	5	3	0	109	30	20	4
BM4	30	5	3	35	109	30	20	4
BM5	100	20	3	82	223	90	200	4
BM6	25	5	18	0	140	30	20	4
BM7	30	10	18	7	23	30	20	4
BM9	250	10	18	7	23	250	90	10
BM10	500	20	18	13	45	-	180	-
BM12	1000	20	3	64	167	-	300	-

Table 1. Benchmark Statistics and Algorithm Parameters (timeouts in seconds).

```
Greedy-Truck-Agent (GTA)()
1  while there exists some commodity to be picked up and demands to be met
2  do if I have some commodity
3        then drop it off at the nearest demand location
4        else  pick up some commodity from the nearest warehouse
5  goto final destination
```

Fig. 8. The Agent-based SCAP Algorithm Simulating the Practice in the Field.

results, we compare our proposed storage models with those of the previous study [1]. Our routing time results also include the solution from the Greedy Truck Agent (GTA) algorithm proposed in [1] which mimics how actual decision makers operate in the field and thus provides a sense of improvement and scale in solution quality. The agent-based algorithm uses the same storage model but builds a routing solution without any optimization. Each vehicle works independently to deliver as much commodity as possible using the algorithm in Figure 8.

Baseline Efficiency Results Table 2 depicts the runtime results for the baseline algorithm in [1]. In particular, the table reports, in average, the total time in seconds for all scenarios (T_1), the total time when the scenarios are run in parallel (T_∞), the time for the storage model (STO), customer allocation (CA), the repository routing (RR), the aggregate fleet routing (AFR), and the fleet routing (FR). The first three fields (T_1, T_∞, STO) are averaged over ten identical runs on each of the budget parameters. The last four fields (CA, RR, AFR, FR) are averaged over ten identical runs for each of the budget parameters and each scenario. Since these are averages, the times of the individual components do not sum to the total time. The results show that the approach scales well with problems with 100 repositories or less. However, benchmark 9 (250 repositories) clearly indicates that the runtime of the storage model has exploded and becomes the dominating factor of the algorithm. Benchmarks 10 and 12 are unsolvable due to memory issues: These models require over 3,000,000 variables.

ASSM Quality and Efficiency Results Table 3 depicts the improvement of our ASSM for the SCAP algorithm. Observe in column STO that ASSM runs about

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	89.89	21.90	39.96	13.01	**0.9293**	**0.4670**	9.257	0.1746	10.057	10.13
BM2	169.1	35.93	66.02	10.47	**0.5931**	**0.2832**	16.67	0.1956	19.26	20.00
BM3	98.58	14.51	61.07	13.79	**0.3557**	**0.1748**	7.225	0.1050	12.04	13.33
BM4	184.2	26.25	68.76	5.163	**0.8892**	**0.3940**	21.24	0.2075	19.58	20.00
BM5	1308	62.01	520.5	32.70	**46.70**	**21.31**	90.87	1.225	128.0	200.0
BM6	723.5	58.76	75.34	3.079	**5.165**	**3.076**	10.81	0.1281	13.35	15.56
BM7	832.0	97.05	75.13	13.31	**16.15**	**5.153**	5.500	0.4509	19.31	20.00
BM9	16123	13661	11108	13459	**10672**	**13458**	143.7	1.377	65.97	90.00

Table 2. Runtime Statistics in Seconds for the Baseline Algorithm (SSM).

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	89.77	22.19	39.25	13.28	**0.5464**	**0.2389**	9.043	0.1791	10.04	10.12
BM2	169.4	35.91	65.93	10.49	**0.4846**	**0.1850**	16.81	0.2084	19.26	20.00
BM3	98.73	14.49	61.15	13.81	**0.3986**	**0.1609**	7.245	0.1092	12.05	13.33
BM4	182.8	24.28	69.74	3.822	**0.6950**	**0.3717**	20.88	0.2122	19.56	20.00
BM5	1266	70.41	487.4	35.18	**18.40**	**7.700**	90.88	0.8691	123.9	200.0
BM6	714.86	59.04	73.28	1.032	**3.130**	**1.041**	10.57	0.09642	13.27	15.56
BM7	823.6	98.79	67.95	12.99	**8.849**	**2.666**	5.479	0.4475	19.28	20.00
BM9	6377	803.6	1184	363.8	**747.7**	**363.5**	153.9	0.8491	66.72	90.00

Table 3. Runtime Statistics in Seconds for the Aggregated Model (ASSM).

twice as fast on benchmarks 1 through 7 and 14 times faster on benchmark 9. The clustering was obtained using ten samples of the k-means algorithm. The sample with the smallest mean sum is used in the clustered storage model. The distances between clusters are calculated on a scenario-by-scenario basis using the average distance between all pairs of points in each cluster. The runtime benefits of the clustering algorithm are largely due to the reduction in the number of variables in the model. Section 5 analyzed the variable reduction and pointed out that the reduction is tightly coupled with the clustering. Due to geographic considerations in these instances, the clustering exhibits great variation from instance to instance and it is important to report the actual reduction in problem size. Table 4 presents the number of variables of the SSM and the ASSM, as well as the lower bound on the number of variables. Observe that the benefits become more significant as the problem size grows and the runtime results confirm this.

Table 5 describes the relative changes in routing times when using ASSM instead of SSM. The quality degradation of the GTA algorithm is also presented to provide a sense of scale. Because the ASSM is a courser approximation of the travel times, some decrease in routing quality is expected. Fortunately, the reduction in quality is not significant and negligible when compared to GTA. It is also surprising that sometimes the clustering model improves the quality of the routing solution. This is a result of the fact that the travel time objective is only approximated in *all* of the stochastic storage models. When there are large distances between nodes, the ASSM meta-edges provide a more accurate estimate of the number of trips needed between two clusters. Unfortunately,

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
SSM	1875	1875	1875	2700	30000	11250	16200	1125000	-	-
ASSM	1116	1206	1248	1470	12246	7344	9576	237420	-	-
Lower Bound	1101	1101	1101	1443	9948	6606	8658	204300	-	-

Table 4. The Number of Variables in the SSM and ASSM.

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
ASSM Change(%)	-0.356	0.0834	-0.108	-0.504	-0.887	3.54	-0.308	2.95	-	-
GTA Change(%)	56.4	43.1	73.7	52.0	64.0	92.2	55.5	259	-	-

Table 5. Degradation of the Routing Times Compared to the SSM Results.

the ASSM still suffers from the same memory issues as the SSM on very large instances and is unable to solve benchmarks 10 and 12.

LSSM Quality and Efficiency Results Table 6 depicts the performance improvement of the LSSM and the consistent reduction in runtime of the storage model (STO), which runs about 1,000 times faster than the SSM on benchmark 9 on the most difficult configuration and 200 times faster on average. Benchmarks 10 and 12 are now in the scope of the solution method. Note that, due to the enormous size of benchmarks 10 and 12, the customer allocation stage does not return a feasible solution within 1000 seconds. To resolve this difficulty, we simply ignore the integer variables, solve the LP relaxation, and round the results. Table 6 indicates that solving the LP relaxation of these problems can take over ten minutes. The MIP solver is also terminated whenever the optimality gap is smaller than 0.05% to stabilize its runtime behavior on the largest instances.

Table 7 describes the relative change in routing times from the SSM. The quality degradation of the GTA algorithm is also presented to provide a sense of scale. Because the LSSM has no information about the travel time, some decrease in routing quality is expected. Again, it is impressive that the reduction is so small (especially when compared with the GTA algorithm). This model is the first that can solve benchmarks 10 and 12, and thus can report the relative improvements over the GTA algorithm. However, it cannot report the relative change compared to the SSM because that model cannot solve these instances. Some policy makers may be concerned by the 7.0% increase in delivery time in benchmark 6 and may prefer to use the SSM. However, some types of disasters require immediate response where every minute is valuable. In those extreme situations, the ASSM and LSSM provide a much faster alternative to the SSM. Our results thus allow policy makers to choose on a case-by-case basis which is preferable: A more immediate response or a higher quality solution.

The lack of information about travel time is an advantage for the memory usage of the LSSM. Only three pieces of the problem specification need to be considered, the repository information, scenario demands, and scenario damages. This resolves the memory issues faced by the other models since the travel times can be handled at the scenario level and not globally. This allows the LSSM to scale to the largest benchmarks. Figure 9 visually summarizes the runtime and quality tradeoffs of the SSM, ASSM, and LSSM. The left graph shows how the

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	94.97	24.38	41.32	12.79	**0.11**	**0.06**	11.46	0.19	9.819	10.00
BM2	169.3	36.04	65.59	10.43	**0.16**	**0.08**	16.86	0.24	19.25	20.00
BM3	98.85	14.31	60.97	13.76	**0.17**	**0.09**	7.28	0.13	12.12	13.34
BM4	183.8	24.26	69.15	3.53	**0.25**	**0.19**	21.27	0.25	19.59	20.00
BM5	1240	69.07	468.6	33.70	**2.12**	**0.86**	90.94	0.81	120.6	200.0
BM6	719.8	56.91	70.46	0.08	**0.35**	**0.09**	11.06	0.09	13.23	15.56
BM7	810.5	100.8	59.54	13.51	**0.70**	**0.15**	5.40	0.37	19.17	20.00
BM9	5859	458.6	488.9	26.32	**53.84**	**27.30**	164.8	0.70	65.92	90.00
BM10	32048	1708	1921	108.9	**17.34**	**22.02**	1385*	8.66	175.7	180.0
BM12	18201	73.11	6146	46.54	**14.12**	**0.22**	5485*	14.28	227.3	300.0

Table 6. Runtime Statistics in Seconds for the Lexicographic Model (LSSM).

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
LSSM Change(%)	3.47	-0.891	0.165	0.919	3.38	7.02	-0.0884	6.77	-	-
GTA Change(%)	61.1	42.5	74.5	53.5	67.0	103	55.5	295	78.5	91.5

Table 7. Degradation of the Routing Times Compared to the SSM Results.

runtime of all the storage models varies as the number of repositories increases. The logarithmic scale illustrates the significant time savings of ASSM and LSSM over SSM. The right graph shows the change in the routing quality when ASSM and LSSM are used instead of SSM. Despite the significant reduction in runtime, the degradation of solution quality is no more than 6% on average.

Behavioral Analysis of LSSM The LSSM ignores the algorithm parameters W_x, W_y, and W_z, and implicitly assumes the field constraint $W_x \gg W_y \gg W_z$. Although the other storage models are more flexible in this regard, all the storage models are configured for this field constraint for the purpose of this study. This means that the storage decisions for the LSSM will be exactly the same as the SSM until all of the demands are met. Once all of the demands are satisfied, the LSSM will degrade because it cannot determine how to use additional funds to decrease the delivery time. However, as the budget increases, it will approach the same solution as the SSM because these solutions correspond to storing commodities at all of the repositories. Figure 10 presents the experimental results on benchmark 6 which exhibits this behavior most dramatically (other benchmarks are less pronounced and are omitted for space reasons). The graph on the left shows how the satisfied demand increases with the budget, while the graph on the right shows how the last delivery time changes. We can see that, as the satisfied demand increases, the routing times of both algorithms are identical until the total demand is met. At that point, the routing times diverge as the travel distance becomes an important factor in the objective, but they re-converge as the budget approaches its maximum and all of the repositories are storing commodities. These results confirm our behavioral expectation. The experimental results also demonstrate that the degradation of the decomposed model is not significant when compared to the choices made by the GTA algorithm, representing the practice in the field.

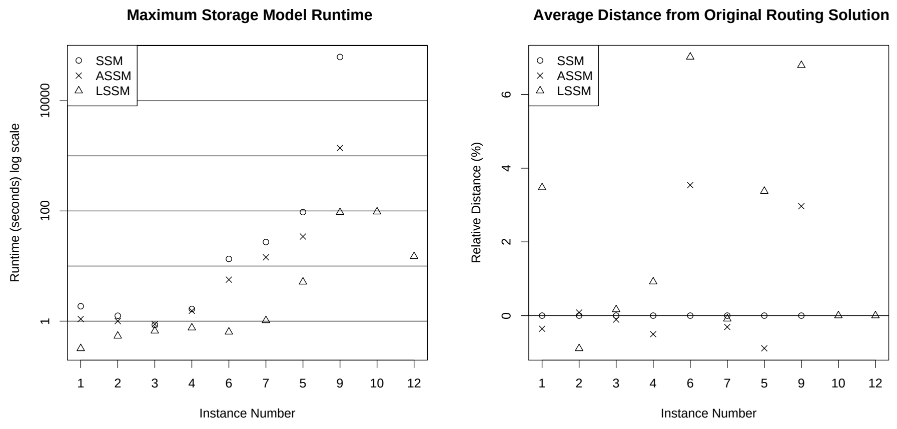

Fig. 9. Runtime and Quality Tradeoffs of SSM, ASSM, and LSSM.

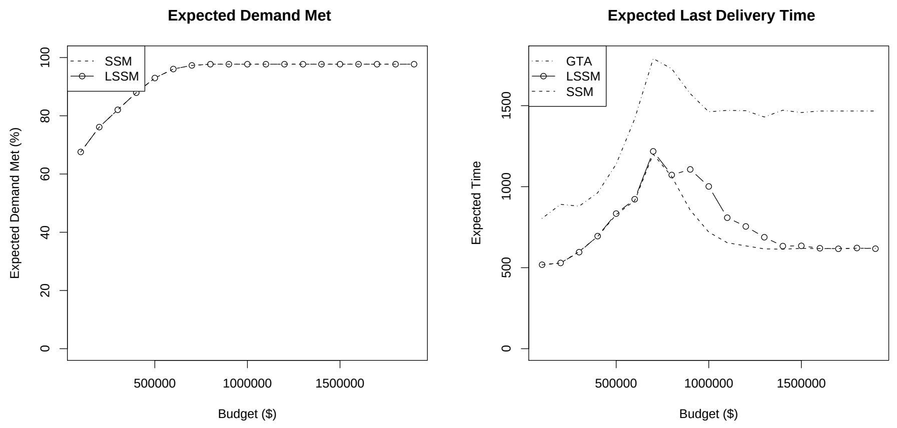

Fig. 10. Behavior of SSM and LSSM on Benchmark 6.

8 Conclusion

This paper studied the scalability of the SCAP problem in the field of humanitarian logistics. The SCAP models the strategic planning process for disaster recovery with stochastic last-mile distribution. The paper proposed two new stochastic storage models that produce high quality solutions to real-world benchmarks that were hitherto unsolvable. The algorithms use spatial and objective decompositions to exploit the problem structure and speedup stochastic storage decisions. The experimental results on water allocation benchmarks indicate that the algorithms are: (1) practical from a computational standpoint; (2) produce significant scalability over previous work; (3) deliver better performance than existing relief delivery procedures. This work is currently deployed at Los Alamos National Laboratory and is activated every time a hurricane of category 3 or above threatens the United States in order to aid federal organizations such as the Department of Energy and the Department of Homeland Security in preparing for, and responding to, disasters.

References

1. Van Hentenryck, P., Bent, R., Coffrin, C.: Strategic Planning for Disaster Recovery with Stochastic Last Mile Distribution. [17] 318–333
2. Wassenhove, L.V.: Humanitarian aid logistics: supply chain management in high gear. Journal of the Operational Research Society **57**(1) (2006) 475–489
3. Beamon, B.: Humanitarian relief chains: Issues and challenges. 34th International Conference on Computers & Industrial Engineering (2008) 77–82
4. United-States Government: The Federal Response to Hurricane Katrina: Lessons Learned (2006)
5. Fritz Institute.: Fritz Institute Website. http://www.fritzinstitute.org (2008)
6. Barbarosoglu, G., Ozdamar, L., Cevik, A.: An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. European Journal of Operational Research **140**(1) (2002) 118 – 133
7. Duran, S., Gutierrez, M., Keskinocak, P.: Pre-positioning of emergency items worldwide for care international. Interfaces (to appear) (2009)
8. Balcik, B., Beamon, B., Smilowitz, K.: Last mile distribution in humanitarian relief. Journal of Intelligent Transportation Systems **12**(2) (2008) 51–63
9. Gunnec, D., Salman, F.: A two-stage multi-criteria stochastic programming model for location of emergency response and distribution centers. In: INOC. (2007)
10. Campbell, A.M., Vandenbussche, D., Hermann, W.: Routing for relief efforts. Transportation Science **42**(2) (2008) 127–145
11. Griffin, P., Scherrer, C., Swann, J.: Optimization of community health center locations and service offerings with statistical need estimation. IIE Transactions (2008)
12. Gunes, C., van Hoeve, W.J., Tayur, S.: Vehicle routing for food rescue programs: A comparison of different approaches. [17] 176–180
13. Toth, P., Vigo, D.: The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Pennsylvania (2001)
14. Ignizio, J.P.: A review of goal programming: A tool for multiobjective analysis. The Journal of the Operational Research Society **29**(11) (1978) pp. 1109–1119
15. FEMA: FEMA HAZUS Overview. www.fema.gov/plan/prevent/hazus (2010)
16. Dynadec, Inc.: Comet 2.1 User Manual. http://dynadec.com/ (2009)
17. Lodi, A., Milano, M., Toth, P., eds.: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010, Bologna, Italy, June 14-18, 2010. Proceedings. In Lodi, A., Milano, M., Toth, P., eds.: CPAIOR. Volume 6140 of Lecture Notes in Computer Science., Springer (2010)