Faster Differentiation of Terrorists and Malicious Cyber Transactions from Good People and Transactions

Peter P. Chen

Foster Distinguished Chair Professor
Computer Science Dept.
Louisiana State University
Baton Rouge, LA 70803, USA
pchen@lsu.edu
http://www.csc.lsu.edu/~chen

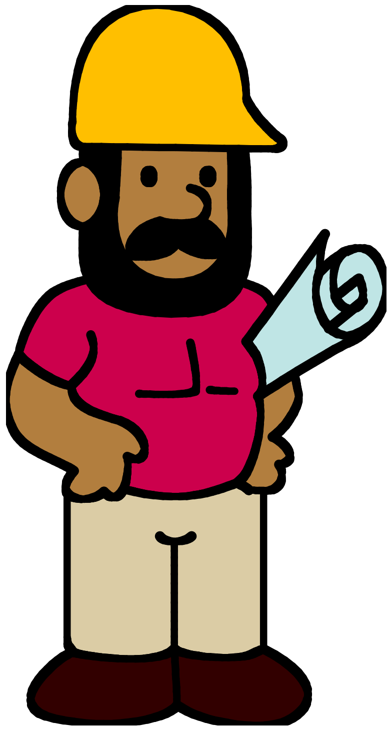
Profiling of terrorists and malicious cyber transactions

- Examples: 9-11, Airport Security, D.C. snipers, Louisiana serial killer, Ohio sniper, etc.
- Current Problems:
  - Isolated Data
  - Questionable data
  - Little Mathematical Analysis
  - Algorithms (if any) are independent of (or incompatible with) data models

Why Do We Study the Profiling Problem?

- 9-11
- D.C. snipers
- serial killers in Louisiana, California, etc.
- Ohio sniper, etc.
- Airport Security

In any population, ...

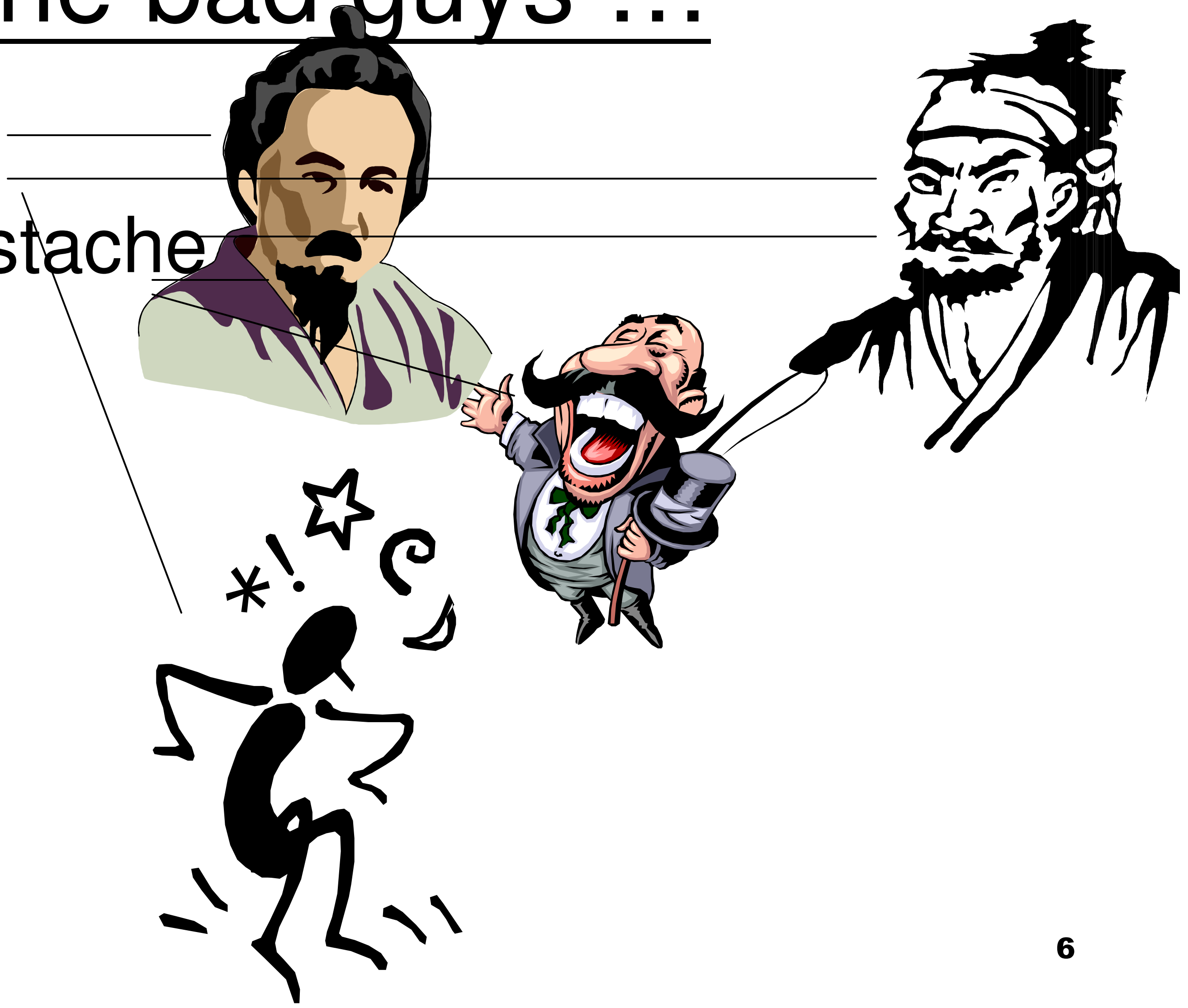


Attributes (and “relationships) of bad guys

- Black hair?
- Beard/moustache?
- Nationality: xxxx?
- Has traveled to Country X three times?

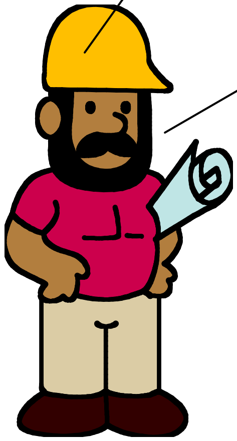
Using the fewest attributes to catch all the bad guys …

- black hair
- beard/moustache



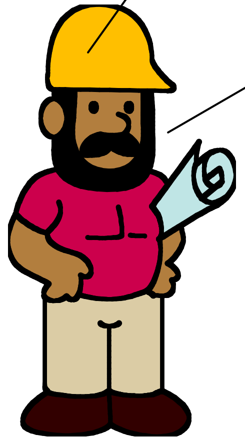
…also catches some good guys (casualties):

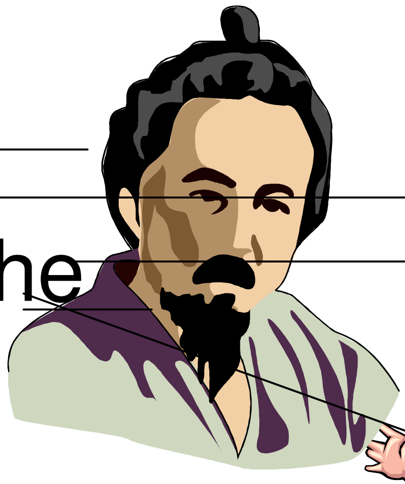
- black hair
- beard/moustache



…also catches some good guys (casualties):

- black hair
- beard/moustache





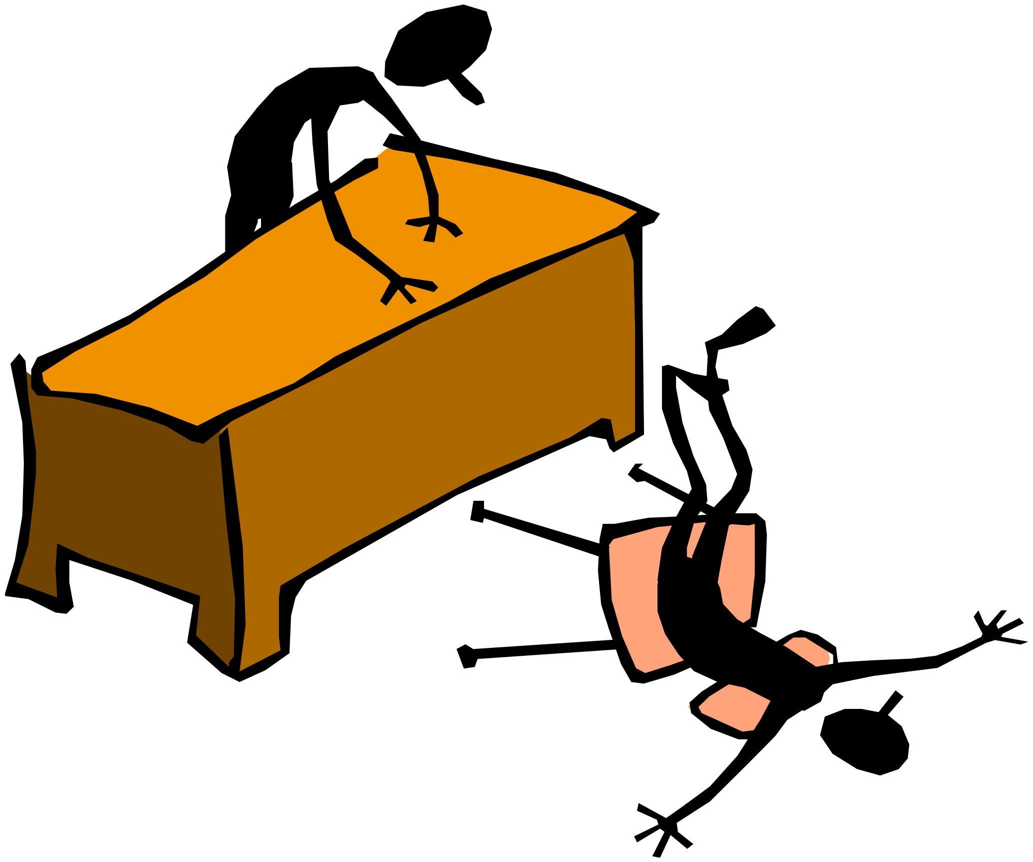
Goal:

• Find the smallest number of attributes that will catch all the bad guys,

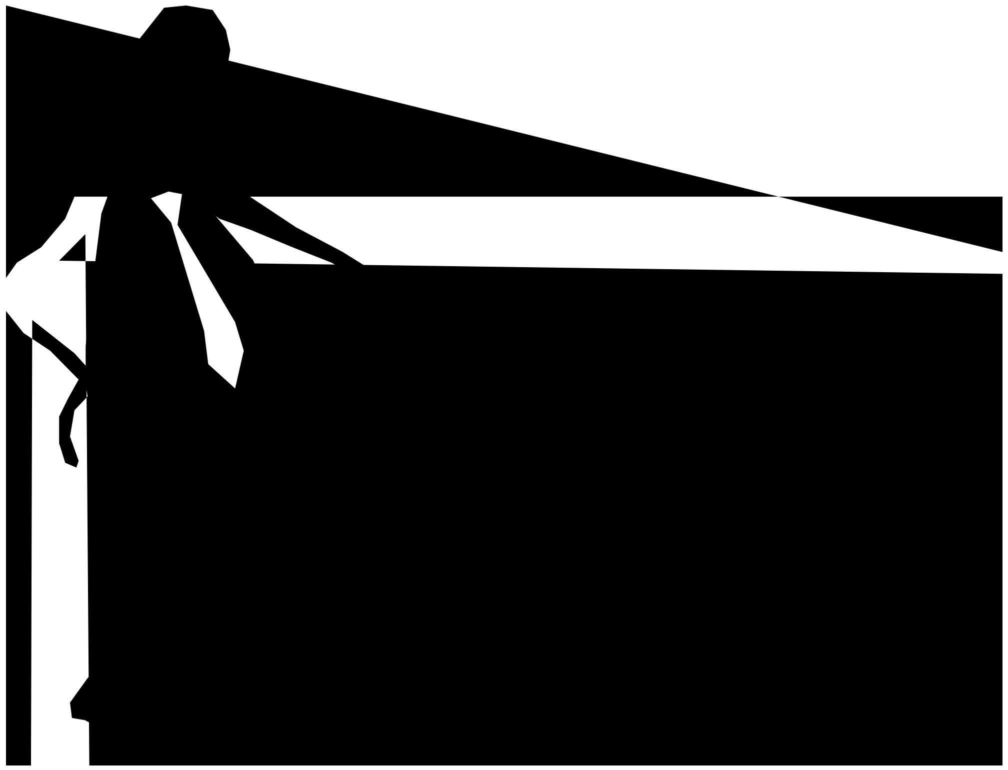
but at the same time

• Include as few casualties (good guys) as possible.

Some good guys are more important than others



Some bad guys are more important (to capture) than others

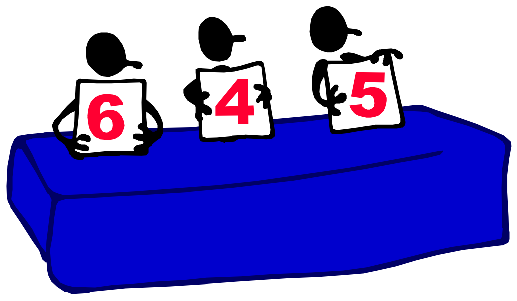


Goal (more ambitious):

- Find the smallest number of attributes that will catch as many, and preferably the more important bad guys,

but at the same time

- Include as few, and preferably the less important good guys, as possible.



Problem -- Profiling of Terrorists and malicious cyber transactions

- Current Problems:
  - Isolated Data
  - Questionable data
  - Little Mathematical Analysis
  - “Unscientific/Unproven” Methods
  - Algorithms (if any) are independent of (or incompatible with) data models
- Solution:
  - Data “links” (“relationships”)
  - Info validity and conflict resolution
  - Optimization model & algorithms
  - Integration of data model and algorithms

Solution Techniques for the Profiling Problem (I) – „New“ Concepts of ERM

- Discovering „Links/Relationships“ from Data in Various Sources (such as DARPA‘s EELD Program)
- „Auto“-construction of „Relationships“
- „Dynamically adjusting“ the weights of relationships
- Validity/Credibility Analysis of Data
  - A Paper was published in InfoFusion 2001, Montreal
  - Algorithm was developed
  - Prototype developed
  - Also, developed machine learning algorithm

Solution Techniques for the Profiling problem (II) – (a) Integration of ERM and Math Models,

(b) Developing New Math Models & Algorithms

- We Model the „profiling“ problem as a „generalized set covering problem“
  - Start with the conventional definition of a „set covering problem (SCP)“
  - Then, define a „weighted set covering problem“
  - Finally, define a „generalized set covering problem“
- We have developed several efficient algorithms for solving this type of problems. Some of them are modified versions of the „greedy algorithm“
- Based on our tests, these new algorithms perform better than other algorithms in the SCP case
- We have also obtained and proved some computational complexity bounds

The Set Covering Problem (SCP)

Notation

For any finite collection of sets $\boldsymbol{X}$, define $\overline{\boldsymbol{X}}$ to be the union of all members of $\boldsymbol{X}$.

Given a finite set B and $\boldsymbol{S} = \{S_1, S_2, \ldots , S_n\}$, where $S_i \subseteq B$, $i \in [1..n]$, we call $\boldsymbol{A} \subseteq \boldsymbol{S}$ a *cover* if $\overline{\boldsymbol{A}} = \overline{\boldsymbol{S}}$.

SET COVERING PROBLEM (SCP) definition:

Given a finite set B, and $\boldsymbol{S}$, a collection of subsets of B, find a minimal cover $\boldsymbol{A}$.

Notation 2

Let $\boldsymbol{S}$ be a finite collection of sets, and given a function $w{:}\boldsymbol{S}\to R_+$, the set of non-negative reals, then for any finite $S' \subseteq \boldsymbol{S}$, define

$$\boldsymbol{w}(S') = \sum_{s\in S'} w(s)$$

WEIGHTED SET COVERING PROBLEM (WSCP) definition:

Given a finite set B, and $\boldsymbol{S}$, a collection of subsets of B, a weight function w: $\boldsymbol{S} \rightarrow R_{+}$, find a cover $\boldsymbol{A}$ with minimum total cost, $\boldsymbol{w}$ $(\boldsymbol{A})$.

GSCP generalizes WSCP in three aspects:

- Each $S_i \in \boldsymbol{S}$ is associated with a weighted set $W_i \in \boldsymbol{W}$, where $\boldsymbol{W} = \{W_1, W_2, \ldots, W_n\}$ and $W_i \subseteq G$, $1 \leq i \leq n$, where G is a finite set.
- Each element $b \in B$ is weighted.
- A combination of weighted elements of B with an additional factor λ enables a relaxation of the covering requirement.

To accommodate the first generalization, we define a weight function *c*: $G \to R_+$. Then, for any finite $W' \subseteq G$,

$$\boldsymbol{c}(W') = \sum_{w \in W'} c(w).$$

For any $\boldsymbol{A} \subseteq \boldsymbol{S}$, define the *cost* of $\boldsymbol{A}$,

$$\boldsymbol{c}(\boldsymbol{A}) = \boldsymbol{c}\left(\bigcup \{W_i : S_i \in \boldsymbol{A}\}\right).$$

To accommodate the second and third generalizations, let d: $B \rightarrow R_+$, and let λ ∈ [0, 1]. Then $\boldsymbol{A} \subseteq \boldsymbol{S}$ is called a λ-*d-cover* of $\boldsymbol{S}$ if

$$\boldsymbol{d}\left(\overline{A}\right) \geq \lambda \boldsymbol{d}\left(\overline{S}\right).$$

GENERALIZED SET COVERING PROBLEM (GSCP) **definition**:

Given B, G, $\boldsymbol{S}$, $\boldsymbol{W}$, d, c, λ, find a λ-d-cover of $\boldsymbol{S}$, $\boldsymbol{A} \subseteq \boldsymbol{S}$, with minimum cost $\boldsymbol{c}\,(\boldsymbol{A})$.

Algorithms for GSCP

Greedy Set Covering Algorithm (GSCA)

Modify Chvátal’s algorithm [Chv79] for SCP to accommodate the generalizing parameters.

```
Algorithm GSCA
Input: $\boldsymbol{S}, \boldsymbol{W}, d, c, \lambda$
Output: $\boldsymbol{A} \subseteq \boldsymbol{S}, d(\overline{\boldsymbol{A}}) \geq \lambda d(\overline{\boldsymbol{S}})$

1. Initialize:
    1.1. $\boldsymbol{A} \leftarrow \emptyset$
    1.2. for $i$ from 1 to $|\boldsymbol{S}|$ do
        1.2.1. $S'_i \leftarrow S_i$
        1.2.2. $W'_i \leftarrow W_i$
2. while $d(\overline{\boldsymbol{A}}) < \lambda d(\overline{\boldsymbol{S}})$ do
    2.1. $i\text{-}min \leftarrow i$: Cost $(\boldsymbol{S}, \boldsymbol{A}, S_i, W_i) = \min[\text{Cost}(\boldsymbol{S}, \boldsymbol{A}, S_j, W_j): S_j \in \boldsymbol{S} - \boldsymbol{A}]$
    2.2. Update:
        2.2.1. $\boldsymbol{A} \leftarrow \boldsymbol{A} \cup \{S_{i\text{-}min}\}$
        2.2.2. for each $S_k \in \boldsymbol{S} - \boldsymbol{A}$ do
            2.2.2.1. $S'_k \leftarrow S'_k - S_{i\text{-}min}$
            2.2.2.2. $W'_k \leftarrow W'_k - W_{i\text{-}min}$
```

```
Algorithm Cost_1
Input: $\boldsymbol{S}, \boldsymbol{A}, S_j \subseteq S, W_j \subseteq W$
Output: $cost$

1. if $d(S_j) = 0$ then
        $cost \leftarrow \infty$
   else if $d(\overline{\boldsymbol{A}} \cup S_j) \geq \lambda d(\overline{\boldsymbol{S}})$ then
        $cost \leftarrow \boldsymbol{c}(W_j) / d(S_j)$
   else
        $cost \leftarrow \boldsymbol{c}(W_j) / (\lambda d(\overline{\boldsymbol{S}}) - d(\overline{\boldsymbol{A}}))$
```

Generous Set Covering Algorithm (GSCGA)

Begin with the entire collection of set covers, and iteratively discard what are determined to be the least favorable covering sets.

```
Algorithm GSCGA
Input: $S$, $W$, $d$, $c$, $\lambda$
Output: $A \subseteq S$, $d(\overline{A}) \geq \lambda d(\overline{S})$

1. Initialize
        1.1. $D' \leftarrow S$
2. do
        2.1. $A \leftarrow D'$
        2.2. $i\text{-}max \leftarrow \max_j$ [Liability $(S, A, W, j)$: $S_j \in A$ AND $d(\overline{D' - S_j}) \geq \lambda d(\overline{S})$]
        2.3. if such an $i\text{-}max$ exists then
                2.3.1. $D' \leftarrow D' - S_{i\text{-}max}$
   while $|D'| < |A|$
```

```
Algorithm Liability_1
Input: $S$, $A$, $W$, $j \in$ N+
Output: cost

1. $cost \leftarrow c(W_j)$
```

```
Algorithm Liability_2
Input: $S$, $A$, $W$, $j \in$ N+
Output: cost

1. $cost \leftarrow c(W_j) / d(S_j)$
```

Super Greedy (Generous) Algorithm

Iteratively fix one element of ***S*** in the solution, then use GSCA (GSCGA) to solve the remainder of the problem.

```
**Algorithm** Super Greedy
**Input**: problem
**Output**: bestCost

1. bestCost $\leftarrow$ □
2. **for** each $s \in \boldsymbol{S}$ **do**
        2.1. partialSolution $\leftarrow s$
        2.2. subproblem $\leftarrow$ Reduced Problem (problem, partialSolution)
        2.3. subProblemSolution $\leftarrow$ Greedy Algorithm (subProblem)
        2.4. currentCost $\leftarrow$ Cost (partialSolution + subProblemSolution)
        2.5. bestCost $\leftarrow$ min [bestCost, currentCost]
```

Democratic Algorithm

Create a “concensus” of the outputs of a set of (heuristic) algorithms, remove elements covered by this concensus, and run the algorithms on the reduced problem, retaining the best results obtained so far.

```
Algorithm Democratic
Input: problem, algorithm_list
Output: bestCost

1. bestCost ← □
2. partialSolution ← Ø
3. do
        3.1. subProblem ← ReducedProblem (problem, partialSolution)
        3.2. outputs ← Ø
        3.3. for each Algorithm ∈ algorithm_list do
                3.3.1. subProblemSolution ← Algorithm (subProblem)
                3.3.2. outputs ← outputs ∪ {subProblemSolution}
                3.3.3. currentCost ← Cost (partialSolution + subProblemSolution)
                3.3.4. bestCost ← min [bestCost, currentCost]
        3.4. concensus ← Concensus (outputs)
        3.5. partialSolution ← partialSolution + concensus
while |concensus| > 0

Algorithm Concensus
Input: outputs
Output: concensus ⊆ outputs

1. concensus ← $\bigcap_{\text{subProblemSolution} \in \text{outputs}}$ subProblemSolution
```

Comparisons of Different Algorithms

Table notation

In Table 1, we applied algorithms to ten instances of the GSCP consisting of 200 rows and 1000 columns; in Table 2, we used the first 25 set covering problems in Beasley's OR Library. Abbreviations used are as follows: D: Democratic algorithm; SG: Super Greedy algorithm; G: GSCA; Gen: GSCGA; v.1 (2): Cost or Liability function 1 (2); Balas: four heuristic functions used in the Randomized Greedy algorithm in [BC96]; Balas (Best of 9): "Best of 9" algorithm used in [BC96]; Balas (Rand Gr): Randomized Greedy algorithm in [BC96] .

Table 1. Outputs to instances of GSCP by various heuristic algorithms

	Best result	D (SG, Gen)	D (SG)	D (G, Gen)	D (G)	SG v1	SG v2	G v1	G v2	Gen v1	Gen v2
gsc1	5361	5361	5361	5827	5722	5676	5809	5827	6067	7097	7378
gsc2	5474	5474	5474	5556	5481	5474	5844	5556	6090	6749	7237
gsc3	5766	5766	5805	5910	5910	5827	6263	5910	6577	7645	8060
gsc4	5351	5351	5351	5666	5499	5351	5451	5738	6047	6684	6505
gsc5	5916	5916	5916	6051	5952	5916	6051	6155	6585	7889	7293
gsc6	5443	5443	5443	5727	5727	5592	5845	5727	6408	6692	6854
gsc7	5138	5138	5181	5138	5232	5181	5324	5232	5324	6606	6172
gsc8	4934	4934	4957	5375	5408	5102	5560	5408	6181	6524	6575
gsc9	5128	5130	5130	5128	5130	5130	5547	5130	6145	6607	6611
gsc10	5180	5180	5232	5399	5327	5232	5400	5399	6186	6457	6726
Total	53691	53693	53850	55777	55388	54481	57094	56082	61610	68950	69411

Table 2. Outputs to instances of SCP by various heuristic algorithms

	Opt imal	D (SG, G, Gen)	D (G, Gen)	D (Balas)	D (Balas, Gen)	Balas (Best of 9)	Balas (Rand Gr)
4.1	429	431	433	434	434	434	432
4.2	512	527	529	529	527	529	524
4.3	516	522	523	531	531	537	532
4.4	494	501	506	505	506	506	504
4.5	512	517	518	518	518	518	518
4.6	560	571	577	580	566	582	573
4.7	430	432	444	447	441	447	445
4.8	492	505	509	522	493	509	508
4.9	641	652	663	663	657	664	666
4.1	514	517	527	523	520	523	521
Tot al	5100	5175	5229	5252	5193	5249	5223
5.1	253	262	269	269	268	269	258
5.2	302	313	317	325	317	318	312
5.3	226	229	232	230	230	230	229
5.4	242	244	245	250	249	247	250
5.5	211	212	212	212	212	214	217
5.6	213	216	225	218	216	216	221
5.7	293	302	306	300	299	301	304
5.8	288	297	305	301	301	305	307
5.9	279	285	292	285	285	285	281
5.1	265	272	275	277	275	275	274
Tot al	2572	2632	2678	2667	2652	2660	2653
6.1	138	141	142	140	142	142	142
6.2	146	153	152	155	152	156	152
6.3	145	148	155	148	148	151	148
6.4	131	136	136	132	131	135	132
6.5	161	172	175	178	178	181	176
Tot al	721	750	760	753	751	765	750
Overall	8393	8557	8667	8672	8596	8674	8626
Ranking		1	4	5	2	6	3

Table 3. Number of basic operations executed by the Democratic Algorithm using various configurations to solve instances of SCP

		D(SG, Gen)	D(G, Gen)	D(Balas)	C(Balas, Gen)	D(Balas, Beasley)
4.1		16	16	12	24	15
4.2		16	16	16	36	25
4.3		16	20	12	24	20
4.4		16	16	16	24	20
4.5		16	16	12	18	20
4.6		16	16	12	30	25
4.7		12	12	12	18	15
4.8		16	16	12	24	35
4.9		32	24	16	36	20
4.1		8	20	12	30	30
5.1		16	16	16	24	25
5.2		20	20	16	24	25
5.3		12	12	12	24	15
5.4		16	16	16	30	20
5.5		12	16	12	18	15
5.6		20	20	12	24	20
5.7		20	20	16	36	20
5.8		12	12	12	18	20
5.9		12	12	16	18	20
5.1		16	16	12	18	15
6.1		28	20	16	24	20
6.2		12	24	20	36	25
6.3		16	12	16	36	25
6.4		12	12	12	18	15
6.5		12	36	16	30	20
	Average	16.00	17.44	14.08	25.68	21.00

Table 5. Output of the Democratic Algorithm using Balas/Carrera and Beasley’s algorithms

	Optimal	D(Balas, Beasley)		Bea90		DYNSGRAD 1		DYNSGRAD 2		Had97	
4.1	429	429	*	429	*	429	*	429	*	429	*
4.2	512	512	*	512	*	512	*	512	*	512	*
4.3	516	516	*	516	*	516	*	516	*	516	*
4.4	494	***494***	*	495		496		494	*	494	*
4.5	512	512	*	512	*	512	*	512	*	512	*
4.6	560	***560***	*	561		561		560	*	560	*
4.7	430	430	*	430	*	430	*	430	*	430	
4.8	492	493		493		492	*	492	*	494	
4.9	641	641	*	641	*	641	*	641	*	641	*
4.1	514	514	*	514	*	514	*	514	*	514	*
5.1	253	***253***	*	255		259		254		254	
5.2	302	304		304		311		307		306	
5.3	226	226	*	226	*	226	*	226	*	226	*
5.4	242	242	*	242	*	244		243		242	*
5.5	211	211	*	211	*	211	*	211	*	211	*
5.6	213	213	*	213	*	213	*	213	*	213	*
5.7	293	***293***	*	294		295		293	*	294	
5.8	288	288	*	288	*	289		288	*	288	*
5.9	279	279	*	279	*	279	*	279	*	279	*
5.1	265	265	*	265	*	265	*	265	*	265	*
6.1	138	***140***		141		142		140		141	
6.2	146	146	*	146	*	156		147		146	*
6.3	145	145	*	145	*	145	*	145	*	145	*
6.4	131	131	*	131	*	132		131	*	131	*
6.5	161	***161***	*	162		170		163		162	

*Optimal value.

Which Algorithm is the best?

- By combining various heuristic algorithms we significantly improve the chances of obtaining even better results.
- **Democratic Algorithm**
  - **Greedy**
  - **Generous**
  - **Super Greedy (Generous)**

Near-Term Research Plans --

- Take advantage of LSU's NCSRT, one of the largest training centers of emergency and anti-terrorism workers
- Test the Models and algorithms with law enforcement agencies and other agencies
- Test the data-model/math-model integration problems with real and quasi-real data sets

Other Related Research Activities

- Integration of conceptual models (ER model, etc.) with databases, math models
- New Machine Learning Techniques
- Trustworthiness of Data and Conflict Resolutions
- (High and low-level) System Architecture and Cyber Security
- Cost/Effective Assessments of Security Techniques -- Making real impacts!