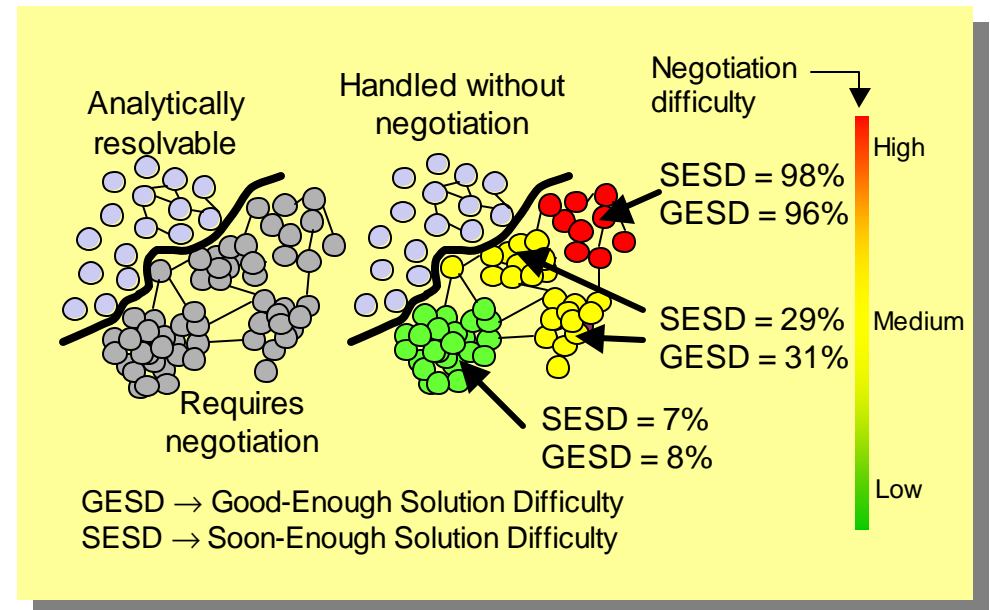


ATTEND
**Analytical Tools To Evaluate
Negotiation Difficulty**

**Computational Complexity Fest
USC ISI - September 6, 2000**

Key Ideas: Difficulty Warnings that Allow Negotiation Systems to Adapt

- **Partition task space** into
 - Those that truly requires negotiation
 - Those that can be solved analytically
- **Identify groups** with different level of difficulty
- **Negotiation difficulty warnings** based on problem complexity
- **Techniques made available** to any negotiation system via a general API



Impact on Negotiations:

Adapt to Time & Quality

- Systems lack analytical methods to **estimate negotiation difficulty** and adapt their behavior accordingly:
 - Can negotiations find any solution in the time available?
→ **Convergence**
 - Can negotiations find solutions with the desired quality and time constraints? → **Closure**
 - Can external changes destabilize the system? → **Stability**

Relation of Techniques Under Consideration to Technical Goals

Identify tasks that truly require negotiations

Form groups with different difficulty level

Generate difficulty warnings by group

API

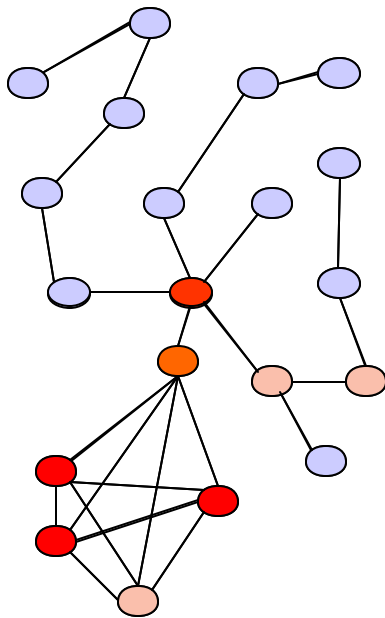
Graph
Coloring,
 K_i -SAT

Phase
Transition
Curves

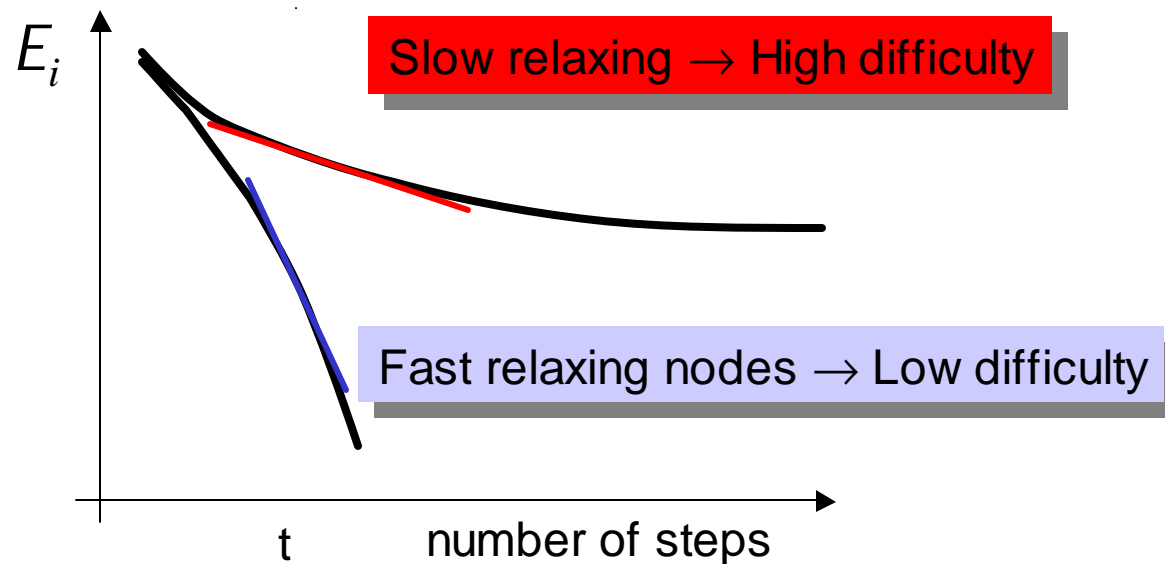
Graph
Partitioning

Statistical Mechanics Approach: Color-flipping Mechanism

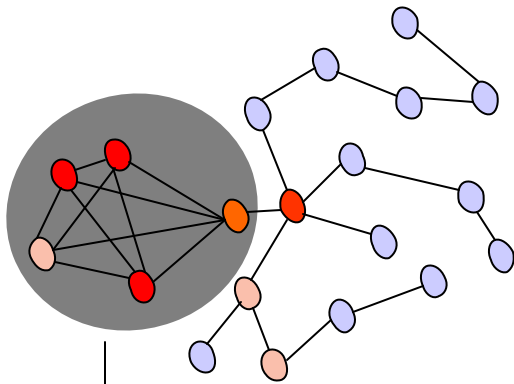
$$E = \frac{1}{2} \sum_{ij} w_{ij} \vec{S}_i \cdot \vec{S}_j$$



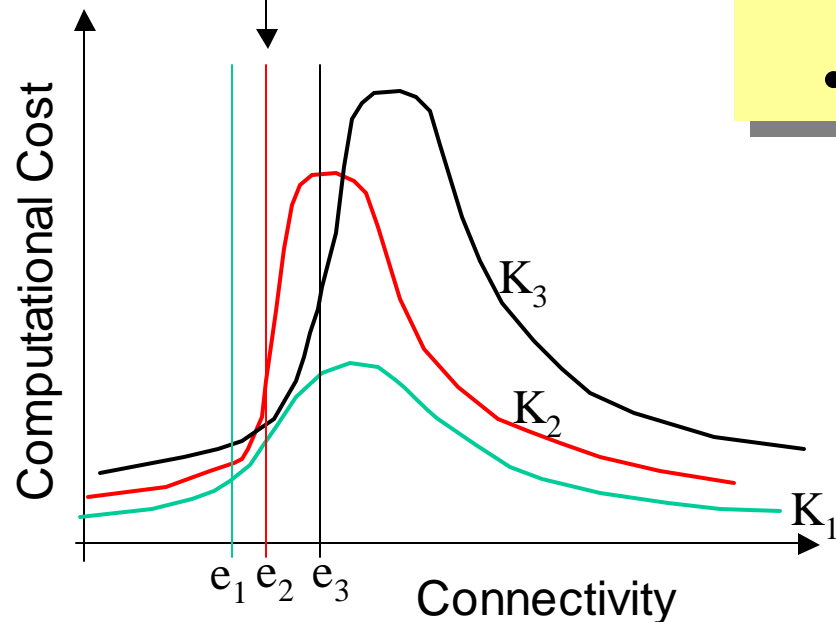
Track the energy relaxation with the number of color flips during minimization using a Metropolis-type algorithm



Phase-Transitions: Problem Difficulty Characterization



- Use transition curves and local value of the order parameter to:
 - Identify groups with different difficulty level
 - Generate difficulty warnings by groups



K_i = number of available colors per node

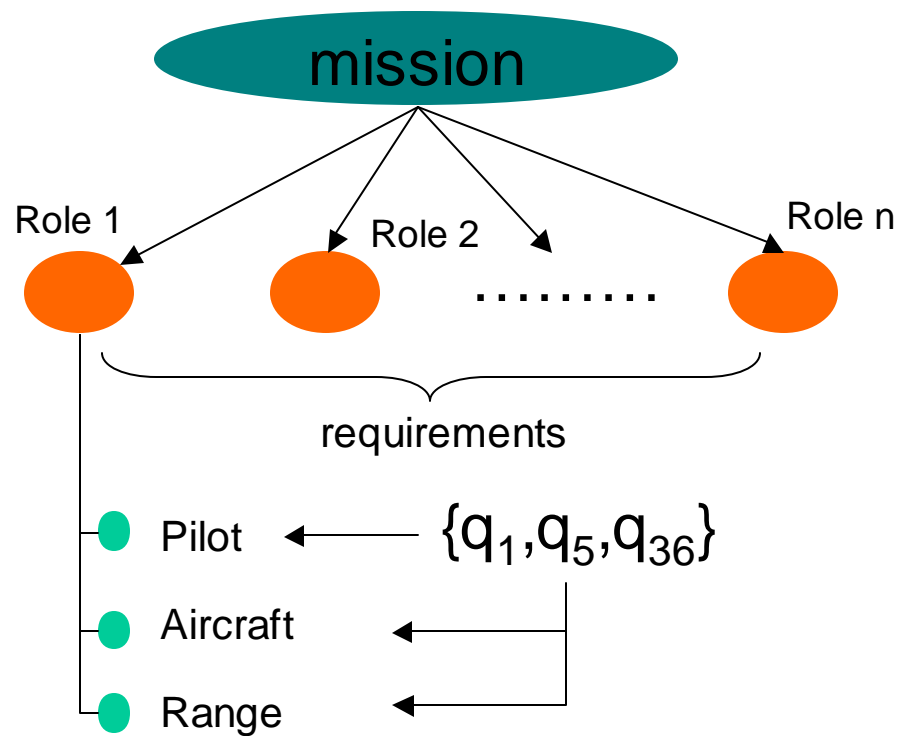
Current Activities

- SNAP scaling empirical studies
- Scaling properties with respect to resources availability
- Run SNAP experiments while varying the pilot, range and aircraft availability and the size of the problem

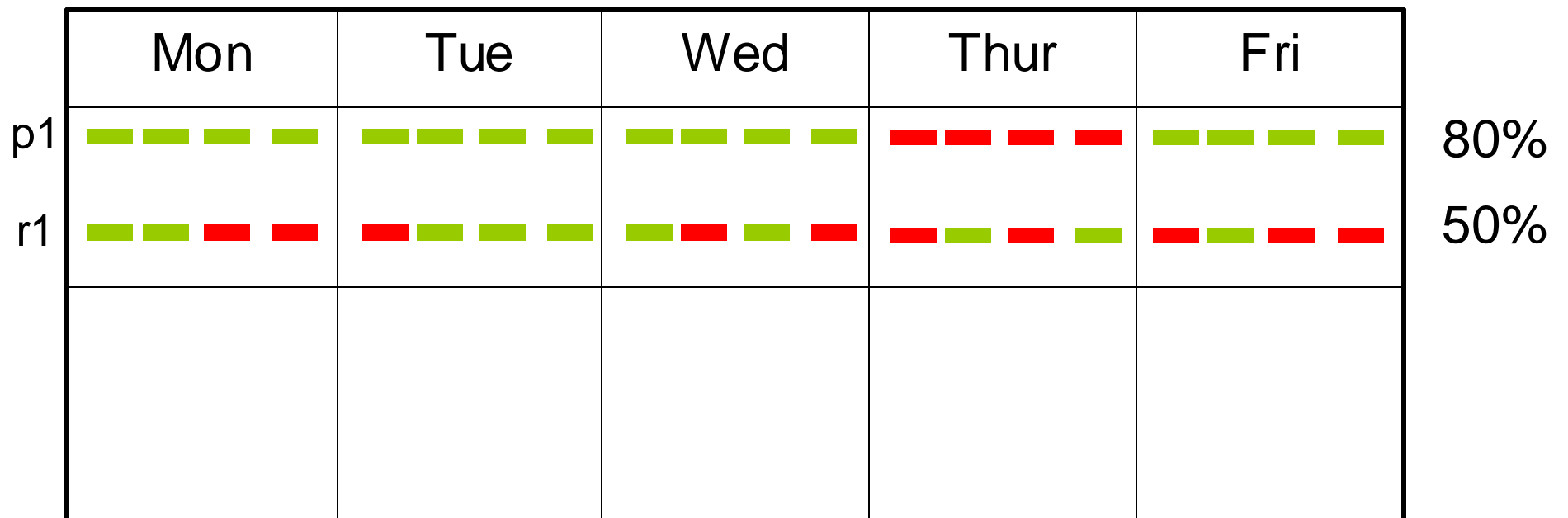
Measurements

- During a run
 - $N(t)$: Number of filled missions at time at which a new mission is filled
- At the end of each trial run
 - Total number of generated missions
 - Resource availability
 - Fraction filled
 - Execution time
- Over several runs
 - Averages of $N(t)$ over several runs

Basic Task/Resources Model in SNAP



Resource Availability: Density Control



↔
range blocking
interval

↔
pilot blocking
interval

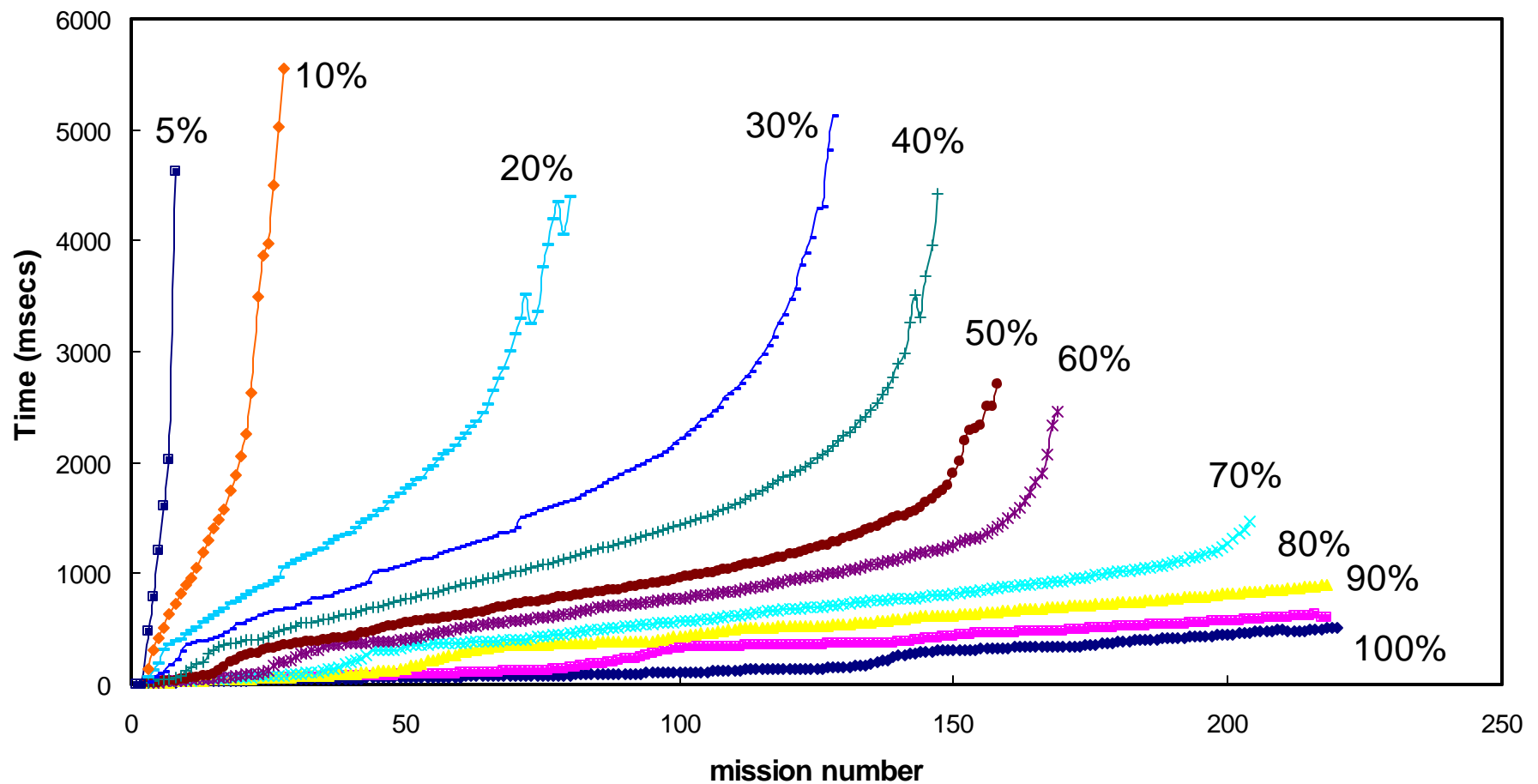
CAMERA-SNAP Experiments Results

■ Experimental procedure:

1. The number of Pilots, Ranges and Aircrafts was kept fixed and their **percentage availability throughout the Planning Horizon was varied randomly (to meet the input availability density) by blocking time intervals of their available time**
2. The size of the problem was controlled by randomly varying the total number of planned missions during the planning horizon (for FRAGS)
3. The planning horizon was kept fixed at 1 week
4. The execution time was measured in batch mode without any GUI cycles taken into account
5. FRAGS only and FRAGS/Training Mix

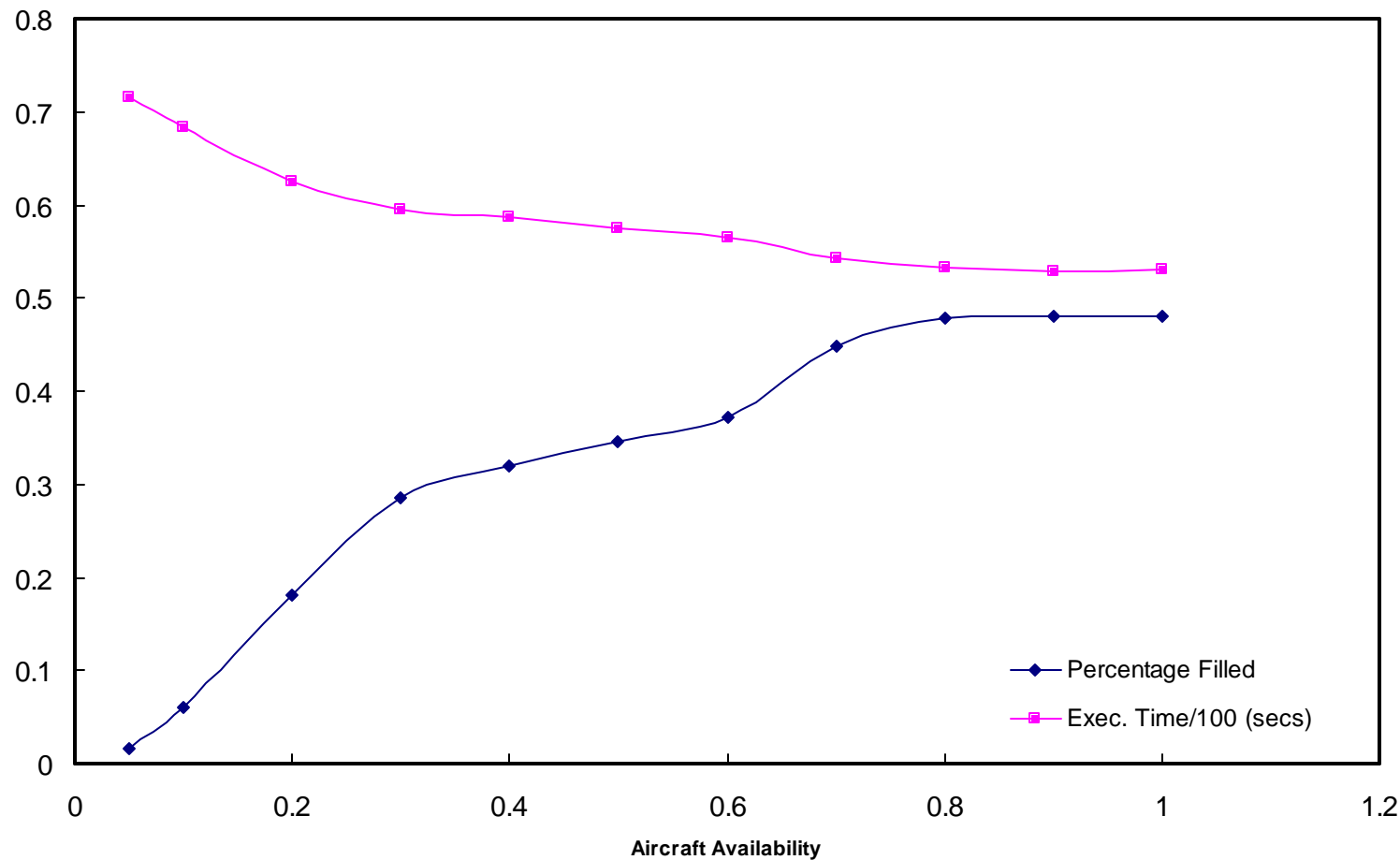
Decreasing Mission Scheduling Rate For Different Aircraft Availability

FRAGS only: 465 missions, 54 pilots, 30 range assignments, 123 aircrafts

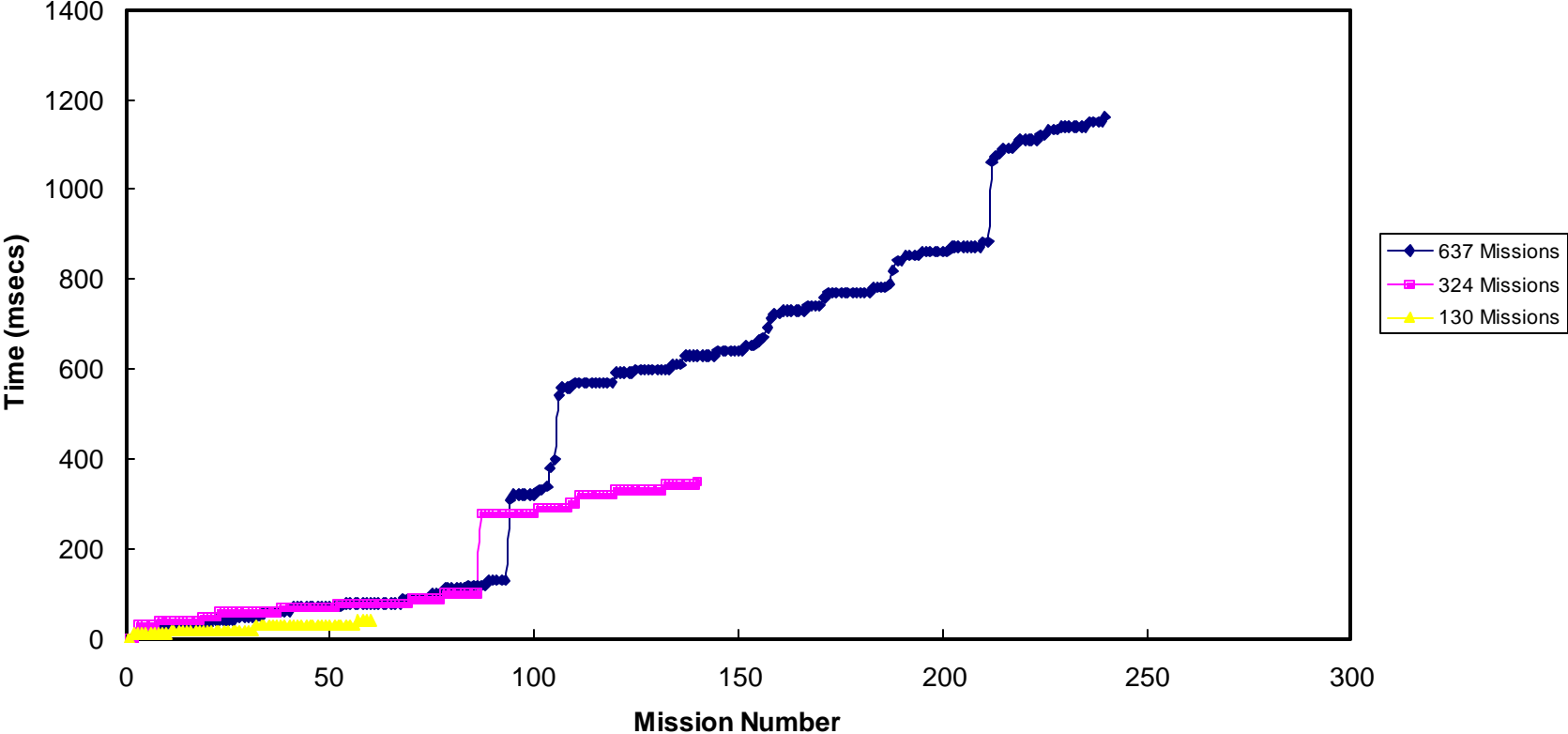


Execution Time and Percentage of Missions Filled For Different Aircraft Availability

FRAGS only: 465 missions, 54 pilots, 30 range assignments, 123 aircrafts

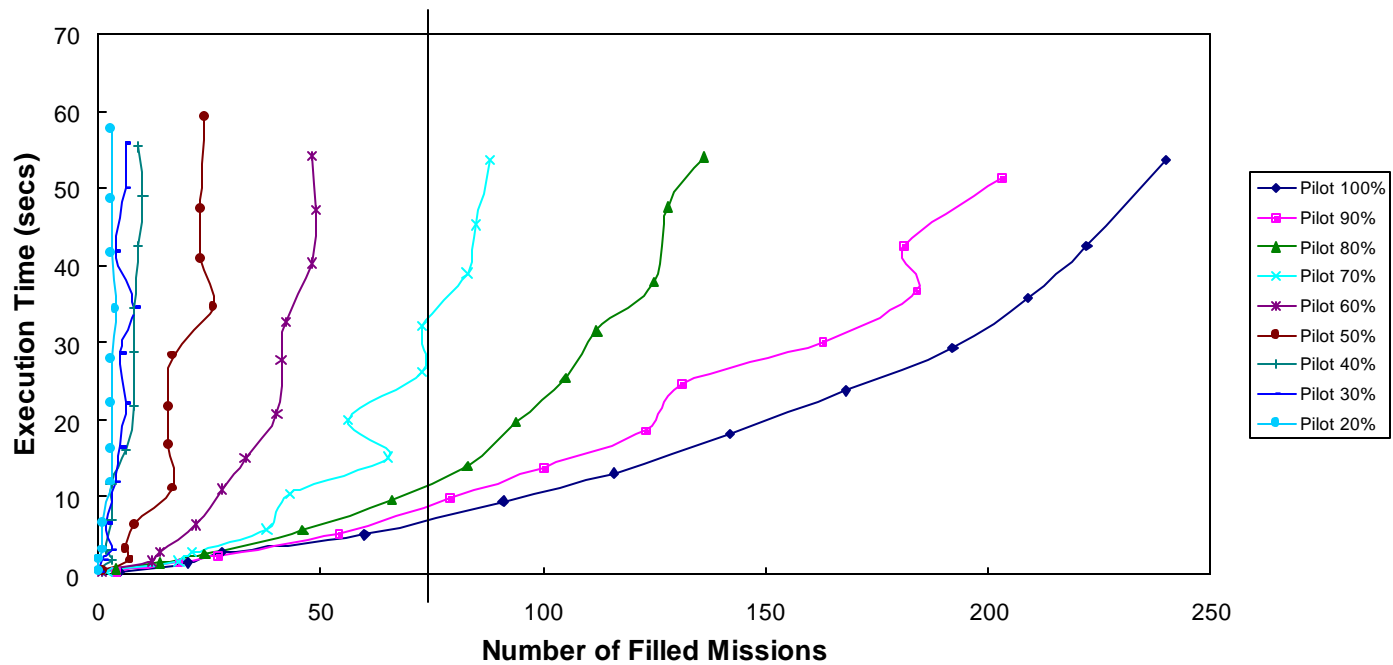


Mission Scheduling Rate For Different Size of the Problem



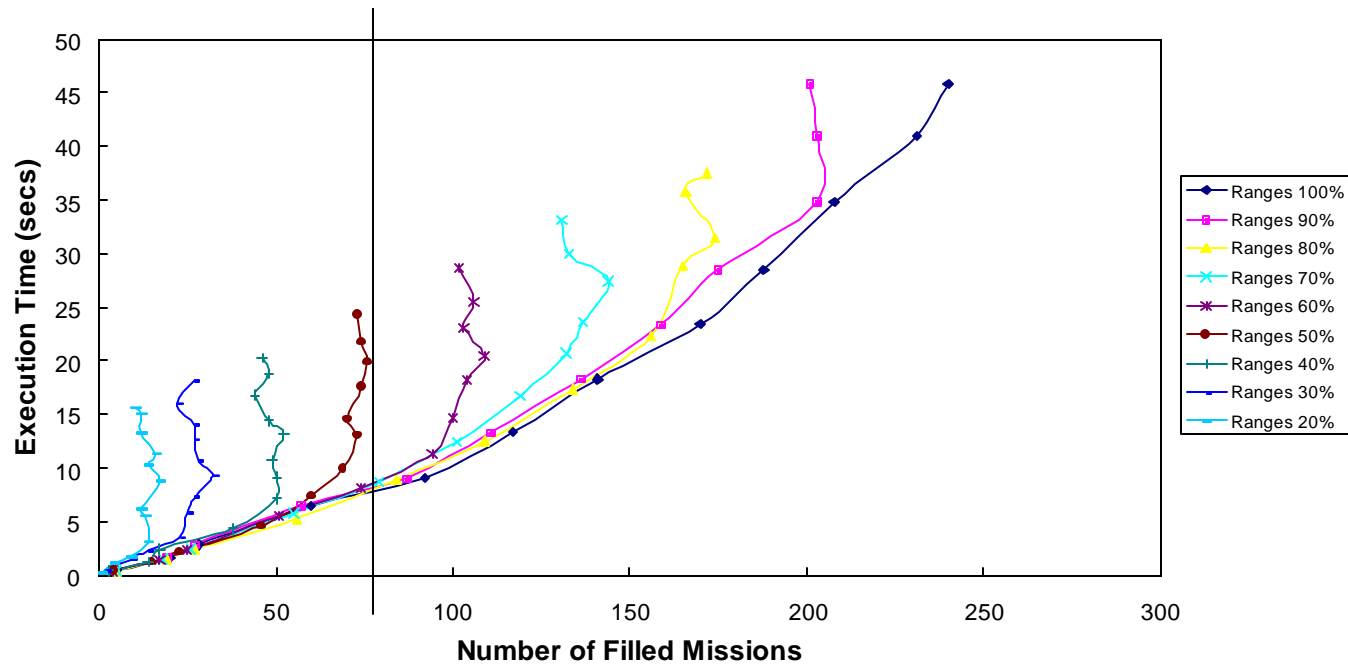
Execution Time vs. Number of Filled Missions

Varying Pilots Availability and Problem size. Ranges and Aircraft Availability fixed at 100%



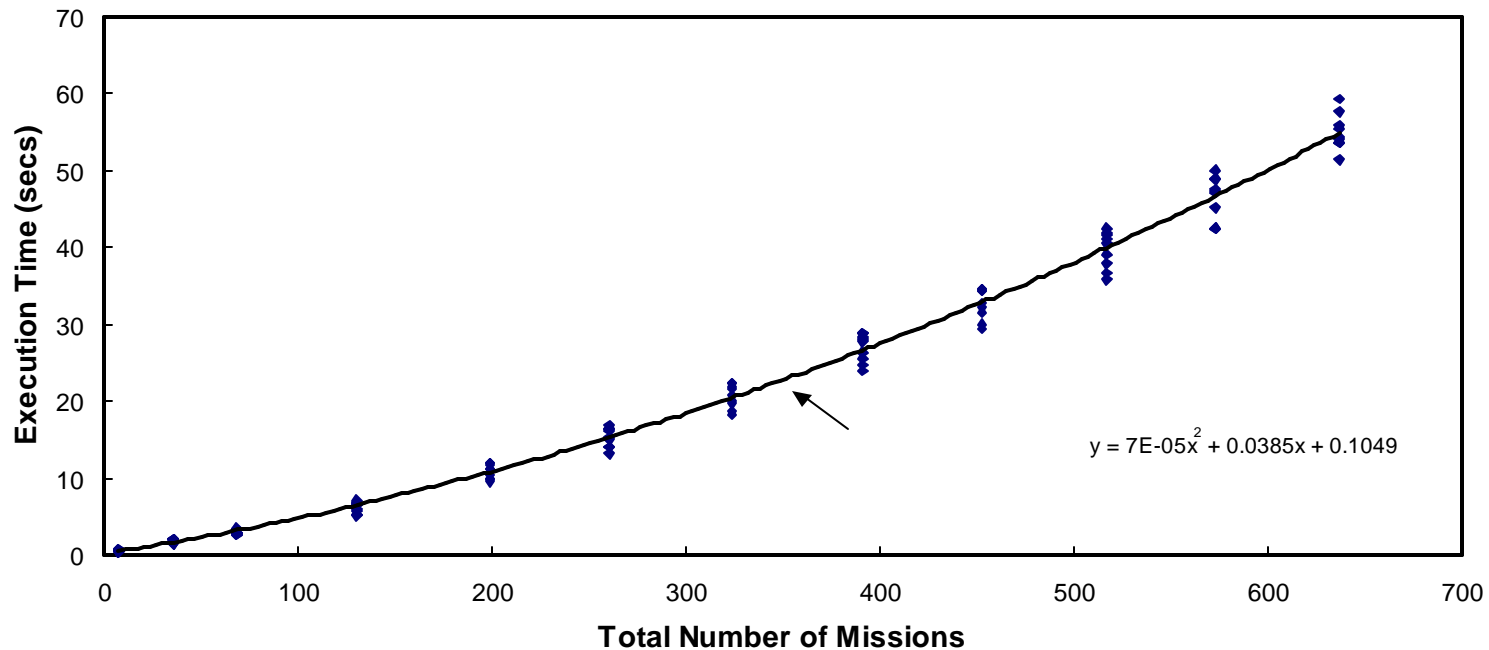
Execution Time vs. Number of Filled Missions

Varying Ranges Availability and Problem size. Pilots and Aircraft Availability fixed at 100%



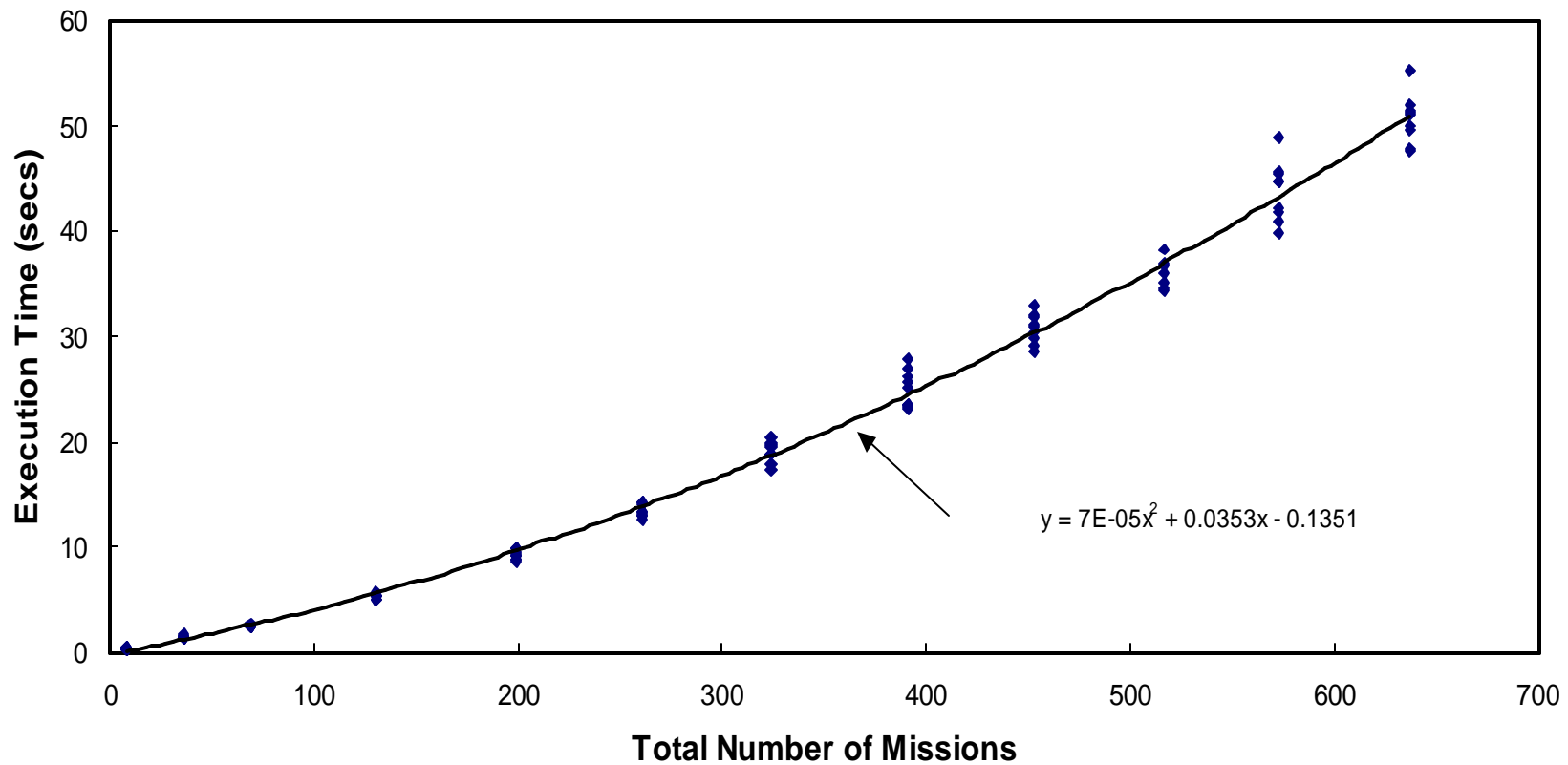
Near Linear Increase of Execution Time wrt the Total size of the Problem (For Pilots)

FRAGS only, Varying Pilots Availability with Ranges and Aircraft Availability fixed at 100%



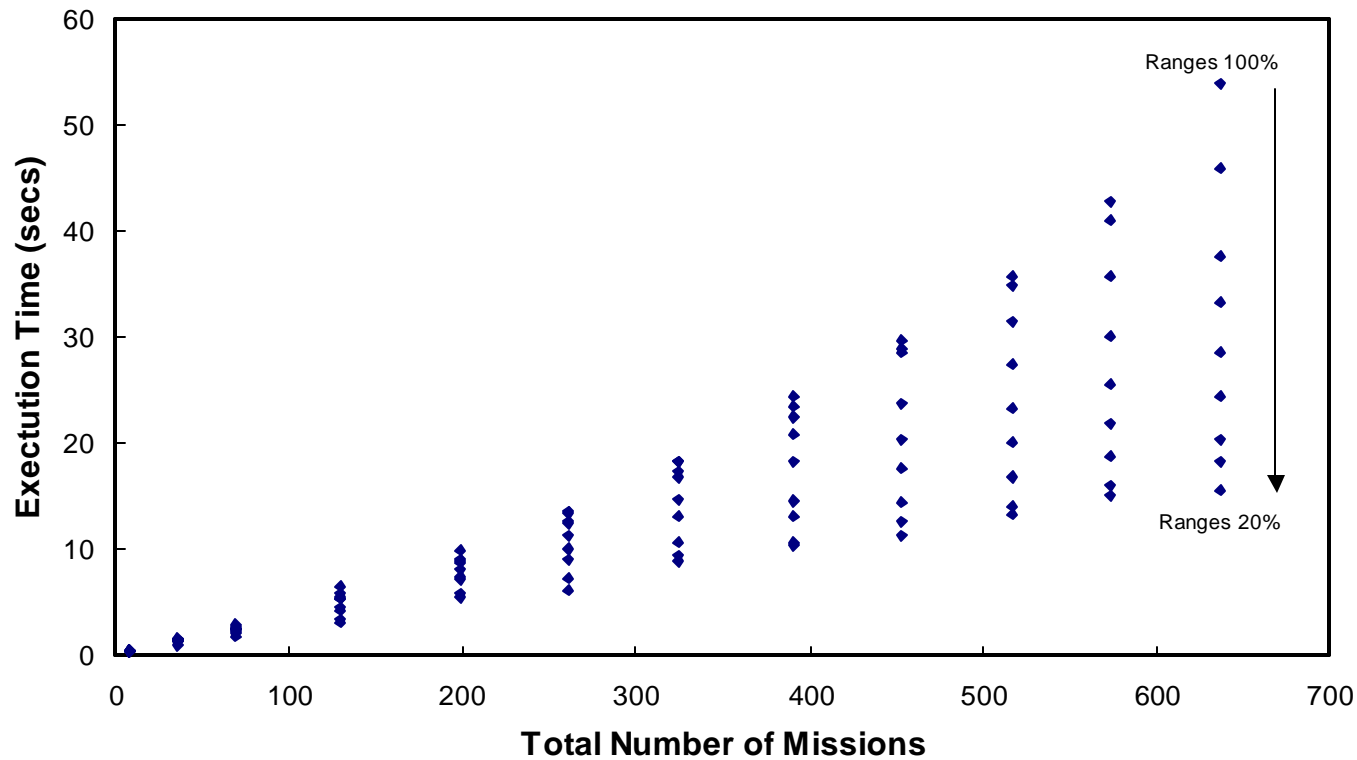
Near Linear Increase of Execution Time wrt the Total size of the Problem (For Aircrafts)

Varying Aircraft Availability with Ranges and Pilots Availability fixed at 100%



Near Linear Increase of Execution Time wrt the Total size of the Problem (For Ranges)

Varying Range Availability with Aircrafts and Pilots Availability fixed at 100%



Experimental Setup

- Run in batch mode with a bash shell script file

```
java -mx128m -ms64m \  
-Ddce.install.path=${DCE_HOME_PCSYNTAX} \  
-Dresource.subdir=\`cygpath -w \`pwd\` \  
edu.isi.dce.na.snap.executive.SnExScalingMain \  
-simulation.trace.output.filename.head xxx-$$ \  
-batch true \  
-randomNumActivitiesWindow 25 \  
-createObserver false \  
-executionPolicy random \  
-displayGui true \  
-displayCommitments true \  
-displaySchedules true \  
-scheduleResolution 30 \  
-displayQuals true \  
-displayMessages false \  
-displayRisk true \  
-displayComputation false \  
-showSortiesGraph true \  
-showCrpGraph true \  
-showNumScheduledActionsGraph false \  
-showPendingActivitiesGraph true \  
-showNumMessagesGraph true \  
-showCommitRatioGraph true \  
-pauseAfterInitialization true \  
-pilotTrainingManualFilename ../pilotmodel/PilotTrainingManual-AV8PQC-Draft.xml \  
-rangeQualsFilename allKnownRanges.xml \  
-rangeAssignmentsFilename rangesvma513ForReal.xml \  
-eventsFilename eventsvma513.xml \  
-aircraftFilename aircraftvma513ForReal.xml \  
-pilotsFilename pilotsvma513ForReal.xml \  
-startTime 2000-07-26T09:30:00Z \  
-sortiesFilename sortiesvma513ForReal.xml \  
-guidanceFilename guidancevma513ForReal.xml \  
-sortieCrawlingDuration 30 \  
-useEquityBasedVersion false \  
-pilotRfcDelay 0 \  
-setupFilename setupForRealAB.csv \  
-pilotRenegesForABetterOffer false $*
```

} Pilots, Aircrafts and Ranges initial availability data

← Setup File

Setup File: .csv file controlled from excel

Setup	Behavior	Detail	Default
x	PilotData	pilotsvma513ForReal.xml	
x	SortieData	sortiesvma513ForReal.xml	
x	GuidanceData	guidancevma513ForReal.xml	
x	RangeDensity		100
x	RangeDensity		50
x	RangeActivityInterval		30
x	AircraftDensity		100
x	AircraftDensity		80
x	AircraftDensity		60
x	AircraftDensity		40
x	AircraftDensity		20
x	AircraftActivityInterval		1440
x	PilotDensity		100
x	PilotDensity		80
x	PilotDensity		60
x	PilotDensity		40
x	PilotDensity		20
x	PilotActivityInterval		1440
x	PilotRenegesForABetterOffer		FALSE
x	PilotRFCDelay		0
x	GridDimension		4
x	NumberOfRuns		500
x	RunDimension		50
x	SortiesFactor		100
x	Run		1
x	Run		2
x	Run		3
x	Run		4
x	Run		5
x	Run		6
x	Run		7
x	Run		8
x	Run		9
x	Run		10

cross product →

Percentage of available Airplanes slots

← Blocking Time Interval in minutes

Number of runs per trial →

← Total number of runs

← Percentage of Initial FRAGS created

Identical Runs

Measurement Mechanism during a SNAP run

- Coding **listeners** one can implement actions at the sending (or receiving) time of a message
- Downcast a message to determine an action based on the **message type**
- Starfields and scaling measurements are implemented using this mechanism
- Problem Generator

Example: Measuring time at which a mission gets filled

```
public void announceMessageAboutToBeSent(
    BaApiParticipant to,
    BaApiParticipant from,
    BaApiParticipant replyTo,
    BaApiMessage x)
{
    if(x instanceof BaApiNegotiationMessage) {
        BaApiNegotiationMessage m=(BaApiNegotiationMessage)x;

        boolean debugAB=true;
        // A sortie announced that it's trying a new time.
        if(m instanceof PrApiRsRequest) {
            PrApiRsRequest r=(PrApiRsRequest)m;
            if(r.getDomainRequest() instanceof SnRrSortieTryingANewTimeSlotTwimcRequest) {
                SnRrSortieTryingANewTimeSlotTwimcRequest req=(SnRrSortieTryingANewTimeSlotTwimcRequest)r.getDomainRequest();
                SnRrSortieTryingANewTimeSlotResource a=req.getSortieTryingANewTimeSlotResource();
                TimeInterval newTime = a.getNewAttempt();
                String id = a.getSortieId();
                computeResourcesAtNewSortieSlot(newTime, id);
            }
        }

        if(m instanceof PrApiCdCommitment) {
            PrApiCdCommitment c=(PrApiCdCommitment)m;
            BaApiDomainResource d=c.getDomainResource();
            if(d instanceof SnApiSortieResource) {
                String id=m.getSenderId();

                tmp = System.currentTimeMillis();
                sNumber++;

                if(pw == null){
                    pw = SnExScalingMain.openFileWriter("schedRate.csv");
                    missionsList = new ArrayList();
                }
                if(SnExScalingMain.runNumber == 1){
                    missionsList.add(new long[SnExScalingMain.maxRunNumber+1]);
                }
            }
        }
    }
}
```

```
--\** SnExExampleListener.mjCode
```

```
(JDE CVS:1.3)--L95--24%-----
```

```
Auto-saving...done
```

Problem Generator

- Generate FRAGS only or FRAGS/Training mix missions
- Generates Pilots, Aircrafts and Ranges
- Their availability is sufficient to fill all generated missions

Outputs

- scaling.csv

Run Number	PilotDensity	AircraftDensity	RangeDensity	TotalNumberOfSorties	TotalNumberOfPilots	TotalNumberOfRangeAssigments	TotalNumberOfAircrafts	SortiesFilled
1	1	1	1	59	18	19	10	47
2	1	0.8	1	44	18	19	10	28
3	1	0.6	1	35	18	19	10	16
4	1	0.4	1	37	18	19	10	13
5	1	0.2	1	34	18	19	10	11
6	1	1	1	70	18	19	10	48
7	1	0.8	1	40	18	19	10	27
8	1	0.6	1	33	18	19	10	16
9	1	0.4	1	44	18	19	10	13
10	1	0.2	1	44	18	19	10	11
11	1	1	1	61	18	19	10	48
12	1	0.8	1	43	18	19	10	28
13	1	0.6	1	35	18	19	10	16
14	1	0.4	1	34	18	19	10	13
15	1	0.2	1	36	18	19	10	11

Outputs

- schedRate.csv

Time (msecs)	mission number
2050	1
2053	2
2063	3
2063	4
2063	5
2073	6
2073	7
2073	8
2083	9
3195	10
3205	11
3355	12
3365	13
3365	14
3385	15
3395	16
3615	17
3886	18
3886	19
4036	20

Outputs

- rateLists.csv

Mission Number	time(msecs)	time(msecs)	time(msecs)	time(msecs)	time(msecs)
1	0	0	0	0	0
2	697	0	0	0	0
3	156	284	223	237	237
4	874	287	257	253	263
5	877	294	263	263	263
6	877	294	270	277	277
7	884	294	273	277	1302
8	887	297	273	277	1659
9	887	297	273	277	2129
10	887	297	797	721	2620
11	1495	727	831	887	3004
12	1519	774	1094	1162	3695
13	1592	858	1231	1525	4833
14	1612	914	1321	2110	
15	1635	995	1449	2984	
16	1679	1085	1779		
17	1692	1225	2046		
18	1812	1368	2266		

The End