

“...80–90% of development cost of a large system is predetermined by the time only 5–10% of the development time has been completed” – INCOSE

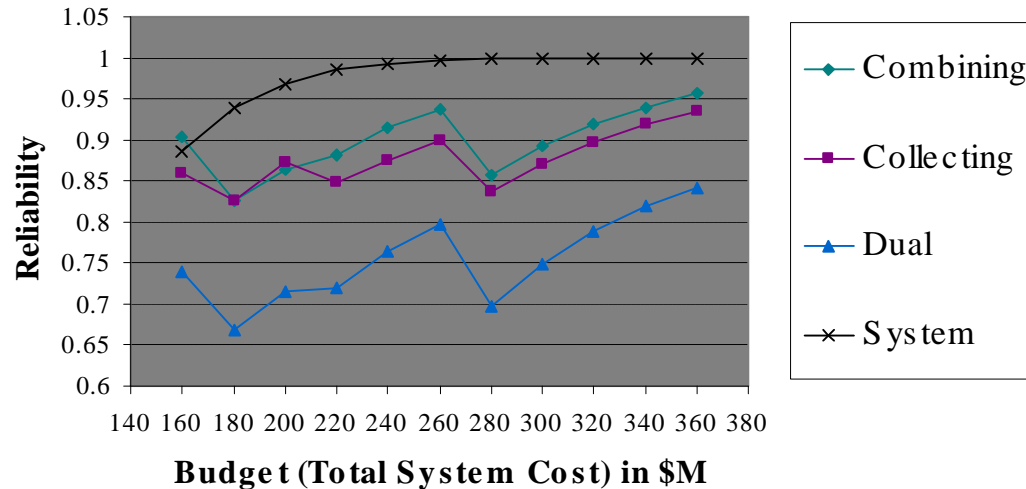
- Think about design decisions based upon lifecycle throughput
 - How many collecting, combining, and dual-functioning spacecraft create the system with the highest productivity for the lowest cost?
 - If there is money for improving the lifecycle throughput of a system, where should it be spent?
 - Increasing redundancy
 - *Add spacecraft to cluster*
 - Increasing reliability of components
 - *Make spacecraft more reliable*

How do you design a system at a high level (cluster size, reliability of spacecraft, etc.) to ensure the highest lifecycle throughput for the lowest cost?

Reliability Optimization Results

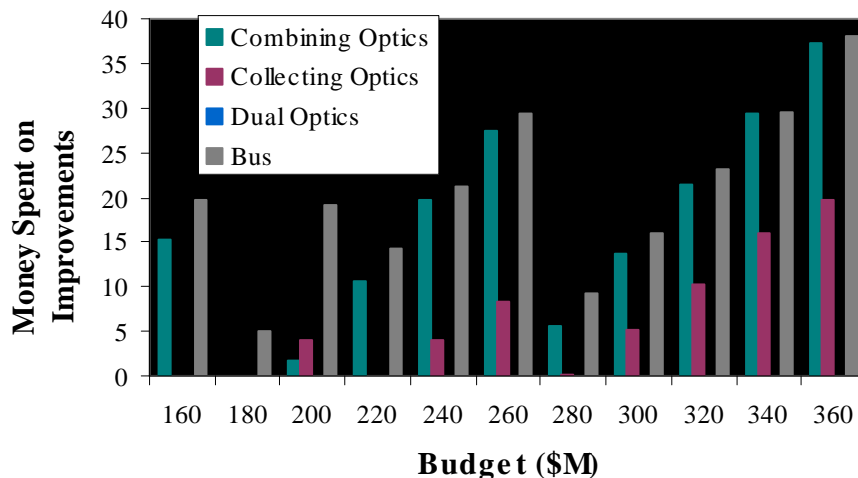


Reliability vs. Budget (Total System Cost)

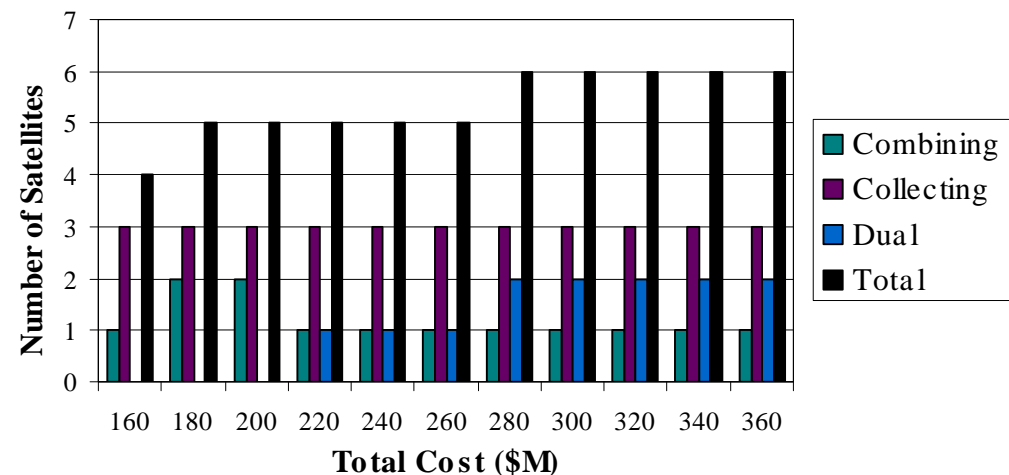


- Total system reliability approaches 1 as budget increases
- In general, improve bus and combining optics first
 - Improving bus reliability improves all 3 types of spacecraft reliability
 - Combining optics generally have less redundancy than collecting optics

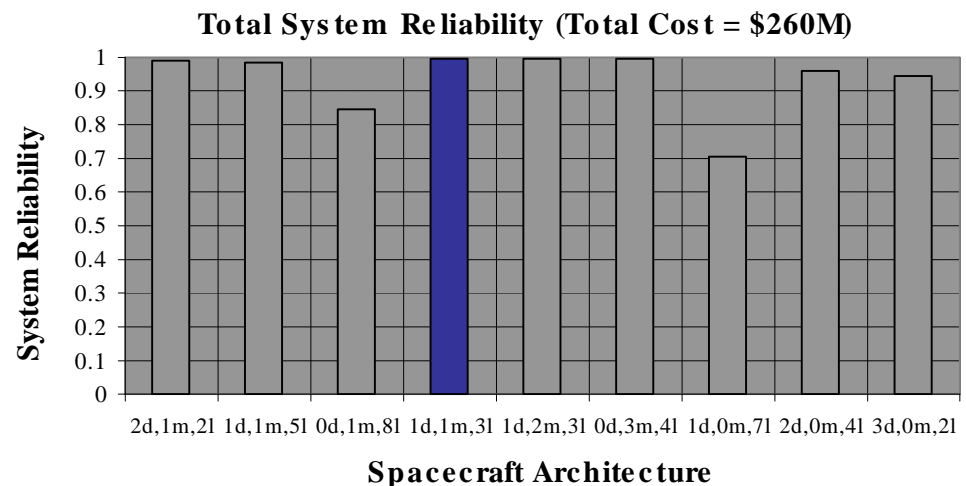
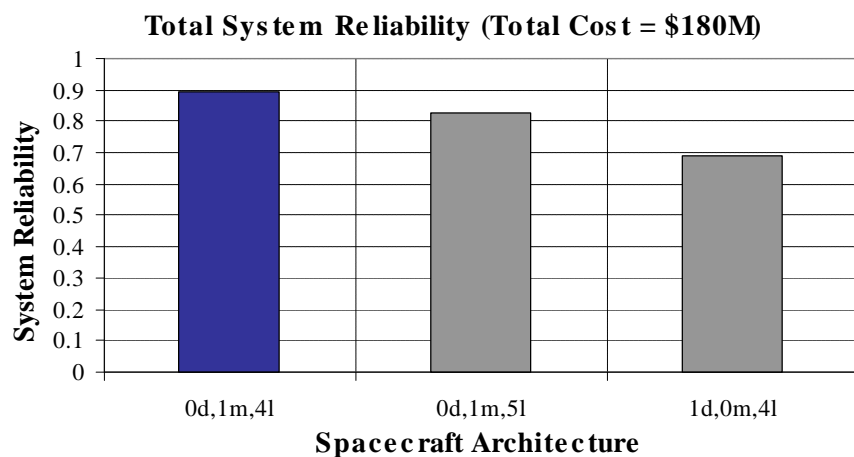
Money Spent to Improve Reliability vs. Budget



Types of Satellites vs. Total Cost



- Compare optimized solutions to other possible solutions
 - Given number of each type of spacecraft, only optimize the amount of the money left over that should be spent on improving each component's reliability
 - Solutions in blue are the reported optimal solutions

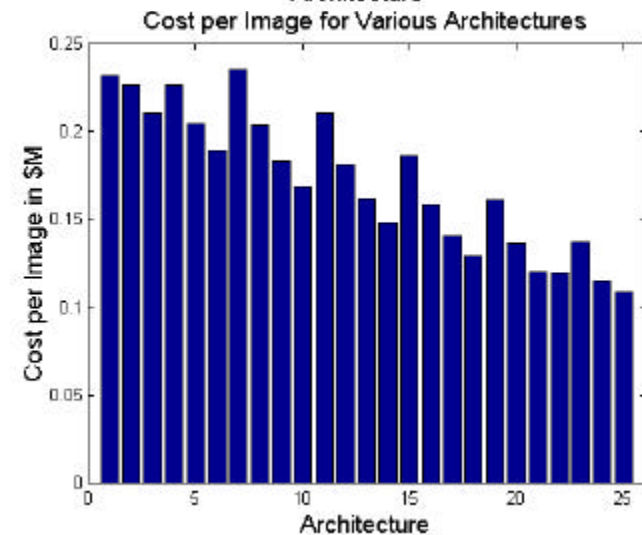
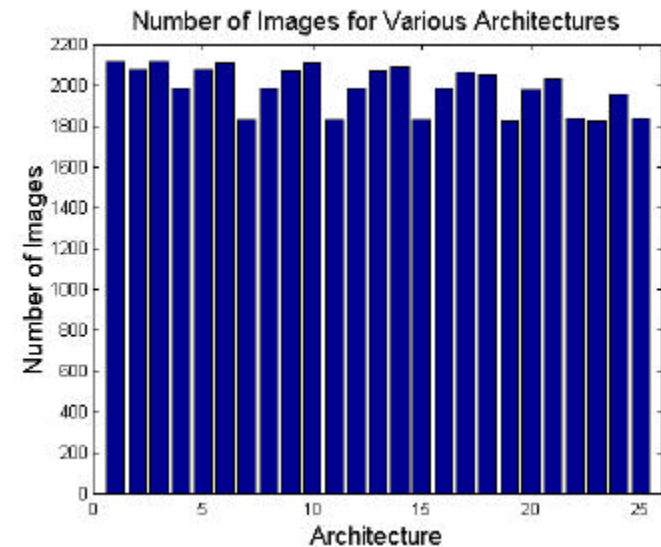


- Optimal answer often has only slightly higher total reliability than other possible answers
 - Need to look at productivity of the systems

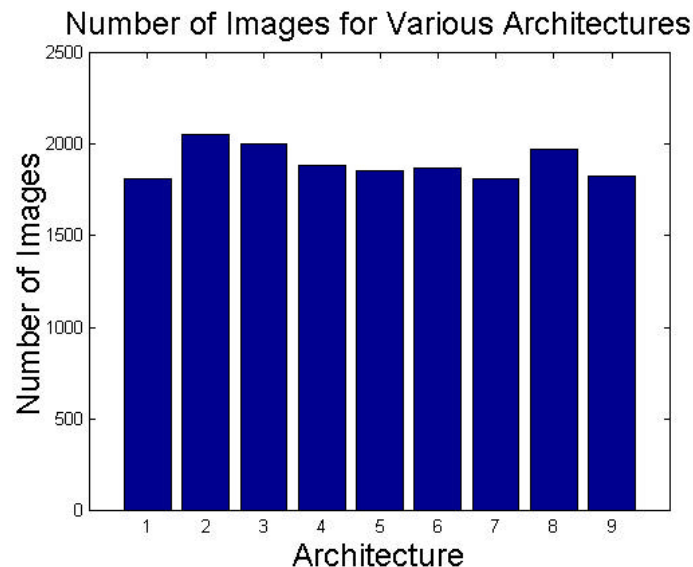
- **Recursive MATLAB function to create tree structure of Markov model**
- **Matrix of state information stores number of each type of spacecraft in each state**
 - Ensures each state is only assigned one state number
 - Keeps track of how many collecting spc. are in each state (n)
- **Method easily applied to different system definitions**
 - Can use any component which could fail instead of spacecraft
 - Can easily change rules for system failure
 - i.e. needs 2 vs. 4 collecting spacecraft to operate
- **Example : All combinations with a total of 7 spc.**
 - Can operate down to 2 collecting spc. + 1 combining spc.

Architecture Key

1. 7 dual, 0 com, 0 col	10. 4 dual, 0 com, 3 col	19. 1 dual, 3 com, 3 col
2. 6 dual, 1 com, 0 col	11. 3 dual, 3 com, 1 col	20. 1 dual, 2 com, 4 col
3. 6 dual, 0 com, 1 col	12. 3 dual, 2 com, 2 col	21. 1 dual, 1 com, 5 col
4. 5 dual, 2 com, 0 col	13. 3 dual, 1 com, 3 col	22. 1 dual, 0 com, 6 col
5. 5 dual, 1 com, 1 col	14. 3 dual, 0 com, 4 col	23. 0 dual, 3 com, 4 col
6. 5 dual, 0 com, 2 col	15. 2 dual, 3 com, 2 col	24. 0 dual, 2 com, 5 col
7. 4 dual, 3 com, 0 col	16. 2 dual, 2 com, 3 col	25. 0 dual, 1 com, 6 col
8. 4 dual, 2 com, 1 col	17. 2 dual, 1 com, 4 col	
9. 4 dual, 1 com, 2 col	18. 2 dual, 0 com, 5 col	



- Find the productivity of the systems analyzed in reliability optimization for a system budget of \$260M
 - Since all have budget of \$260M, architecture with the highest number of images will have the lowest cost per image
 - Reliability of each type of spacecraft entered individually for each architecture such that all total system costs = \$260M



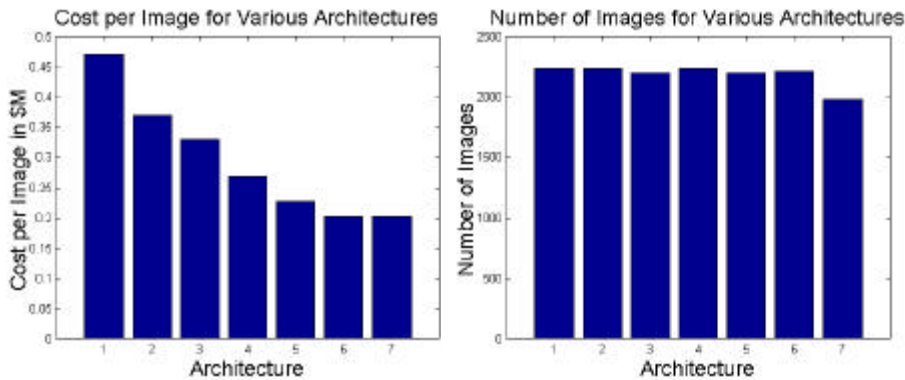
Architecture Key

1. 2 dual, 1 com, 2 col
2. 1 dual, 1 com, 5 col
3. 0 dual, 1 com, 8 col
4. 1 dual, 1 com, 3 col
5. 1 dual, 2 com, 3 col
6. 0 dual, 3 com, 4 col
7. 1 dual, 0 com, 7 col
8. 2 dual, 0 com, 4 col
9. 3 dual, 0 com, 2 col

- Architecture 2 (1 dual, 1 com, 5 col) has the highest productivity, but does not have the highest reliability (Architecture 4)

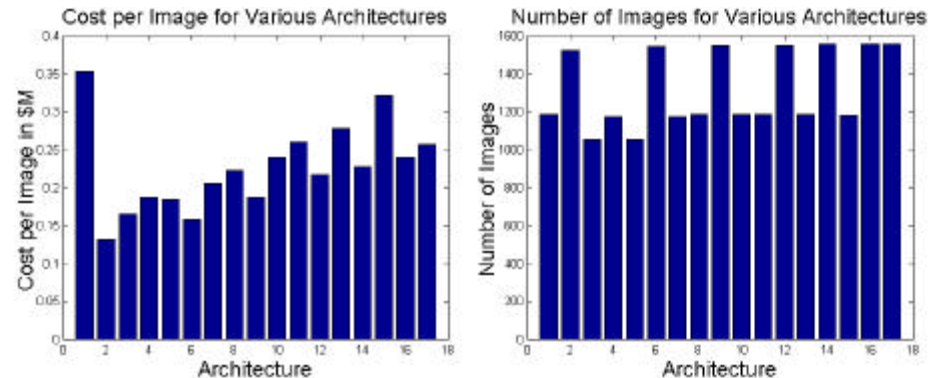
Case Studies (II)

- 6 combinations of 15 total spc.
 - System operational down to 4 collecting spc. + 1 combining spc.
 - Very large and complex system - would take large amount of time to create A-matrix by hand
- All combinations of 6 total spc.
 - System operational down to 2 collecting spc. + 1 combining spc.
 - Requires one combining spc. for every 2 collecting spc.
 - Redefine # of baselines as # of 2 col., 1 com. sets



Architecture Key

- 1. 15 dual, 0 com, 0 col
- 2. 10 dual, 0 com, 5 col
- 3. 5 dual, 5 com, 5 col
- 4. 5 dual, 0 com, 10 col
- 5. 0 dual, 5 com, 10 col
- 6. 0 dual, 3 com, 12 col
- 7. 0 dual, 1 com, 14 col



Architecture Key

- | | | |
|-------------------------|--------------------------|--------------------------|
| 1. 6 dual, 0 com, 0 col | 7. 2 dual, 0 com, 4 col | 13. 4 dual, 0 com, 2 col |
| 2. 0 dual, 2 com, 4 col | 8. 2 dual, 1 com, 3 col | 14. 4 dual, 1 com, 1 col |
| 3. 0 dual, 1 com, 5 col | 9. 2 dual, 2 com, 2 col | 15. 4 dual, 2 com, 0 col |
| 4. 1 dual, 1 com, 4 col | 10. 3 dual, 0 com, 3 col | 16. 5 dual, 0 com, 1 col |
| 5. 1 dual, 0 com, 5 col | 11. 3 dual, 1 com, 2 col | 17. 5 dual, 1 com, 0 col |
| 6. 1 dual, 2 com, 3 col | 12. 3 dual, 2 com, 1 col | |