

Optimizing Dynamic Resource Allocation

**Lucas Krakow, Louis Rabiet, Yun Zou,
Guillaume Iooss**

Sanjay Rajopadhye & Edwin Chong
Colorado State University

Outline

- **UAV Resource Allocation**
 - POMDP Formulation & Non-myopic Belief-state Optimization (NBO)
 - GPU Acceleration
 - Algorithmic advances
- **Resource allocation for polyhedral programs & beyond**
 - Dynamic dependences (Alphabets)
 - Non-polyhedral iteration spaces (while)
 - Dynamic resources for polyhedral programs
 - CART: Constant Aspect Ratio Tiling

UAV Resource Allocation

- Dynamic constraints
- Uncertain and stochastic
- Spatially varying measurement errors
- Data fusion and geometric synergy
- External factors
 - Obstacles (may also act as occlusions)
 - Wind
 - Aggression & evasion

Non-myopic Dynamic Control

- Problem is inherently dynamic
 - Must exploit feedback
 - Poor control actions at one time will lead to regret in the future
- Non-myopic: cannot just apply control action that optimizes performance at that time instant
- Aligned with DDDAS goals

Solution methodology

- Partially Observable Markov Decision Processes (POMDP)
- Solved using approximation method called NBO (nominal belief-state optimization)

GPU Acceleration

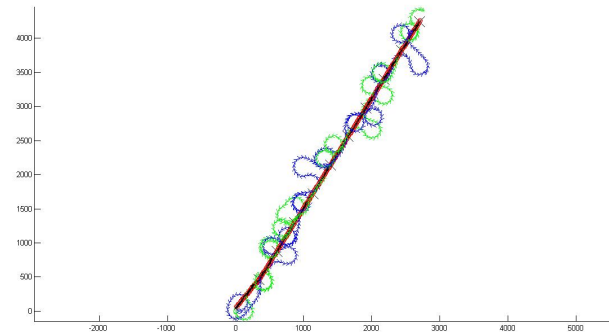
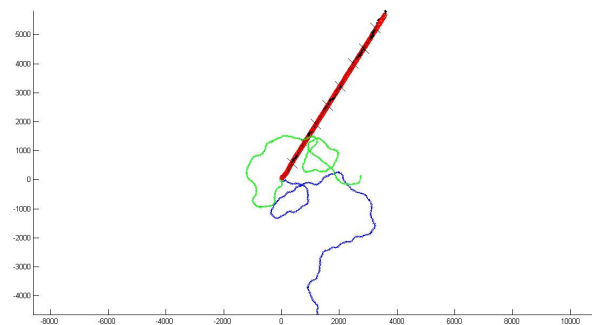
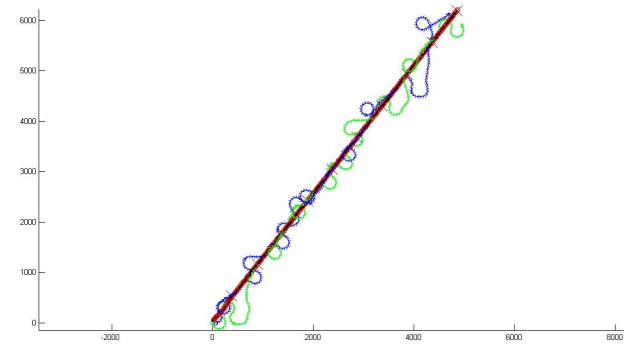
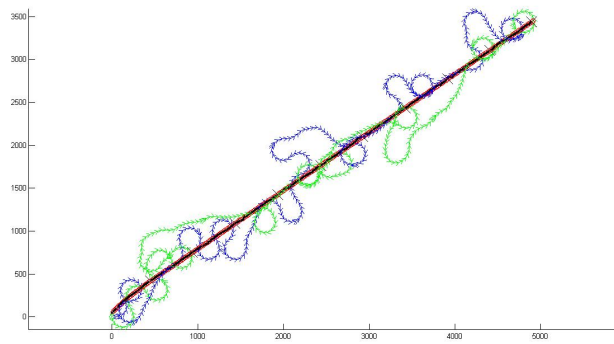
- Initial prototype: Matlab implementation using the **fmincom** library function.
- Main computational bottleneck: repeated calls to evaluate **objfuntrace**, the objective function to evaluate candidate solutions
 - 70% time, but in several thousand calls
- Solution: parallelize **objfuntrace** at fine grain, and also replace **fmincom** by an alternate coarse grain parallelization:
 - Nelder-Mead
 - Particle Swarm Optimization (PSO)

Fine grain parallelization

- Individual calls to `objfuntrace` have relatively small matrices and time horizons
- Must parallelize many independent calls to fully exploit GPU functionality
- 6-D iteration space, hand parallelized
 - Memory-parallelism tradeoffs (at all levels)
 - Matlab \rightarrow C \rightarrow CUDA (speedup = 2×10^3)
 - Mainly (2+ orders of magnitude) in Matlab \rightarrow C
 - interpreted vs compiled
 - better memory management
 - 5-10x speedup in C \rightarrow CUDA

Coarse grain

■ Numerical precision: **fmincom** vs **PSO**



Algorithmic advances

- Extension for data association:
Multi-Hypothesis Tracking. Need to modify
 - State & State transition law
 - include tracking state
 - Observation & observation law: MHT includes false alarms, missed detection, etc.,
 - Use a probabilistic model
 - Cost function
 - additional term for target ambiguity
 - Belief state
 - Distribution over states (one term is unobservable and updated via Bayes theorem – Kalman filter MHT)

Outline

- UAV Resource Allocation
 - POMDP Formulation & Non-myopic Belief-state Optimization (NBO)
 - GPU Acceleration
 - Algorithmic advances
- Resource allocation for polyhedral programs & beyond
 - Dynamic dependences (Alphabets)
 - Non-polyhedral iteration spaces (while)
 - Dynamic resources for polyhedral programs
 - CART: Constant Aspect Ratio Tiling

Dynamic polyhedral programs

- Extend the expressivity:

Alphabets: an extension of the polyhedral equational language Alpha

- Iterative termination through **unbounded polyhedra** & **fixed-point semantics**
- Non-affine dependences through uninterpreted functions
- Rework **mathematical closure** properties

Compiling Alphabets

boolean cond(t); //cond will be evaluated at every iteration

affine counter {N | N>2} over {t|t>0} while cond(t)

inputs

float Init{i|0<i<N};

dep Z {n->j | 0<i<N && 1<=j<=2};

outputs

float Y {n|0<n<N};

let

Y[i,t] =

case

{|t==0}: Init[n,0];

{|t<=n}: Y[n,0];

{|t>n && n==0}: Y[n,0];

{|t>n && n>0 && Z[n]==2}: Y[n,t-n]-1;

{|t>n && n>0 && Z[n]==1}: Y[n,t-n]+1;

esac;

Compiling Alphabets

Alpha(bets) is equational/declarative (single assignment). Compiler analyses:

- Scheduling
- Lifetime (memory allocation)

Static scheduling is undecidable

[SQ'89]

fallback strategy: demand-driven evaluation

- **Alphabets**: unbounded computations →
(potentially) unbounded memory

Optimizations

- Memory bound analysis
 - To determine the maximum amount of history that needs to be stored – provably bounded, but may be a function of program size parameters
- Speculative evaluation
 - When the termination condition is monotonic:
 - if $\text{cond}(t)$ becomes false for some t , it implies that for any $t' > t$ the condition $\text{cond}(t')$ is also false
 - No harm in advancing t by a extra iterations and then correcting as necessary (needs “check-pointing”)

Changing target architectures

“JIT” compilation of polyhedral programs:

Maximize static compilation, leave as many “tunable” parameters.

- Challenge: hierarchical & parametric tiling:
(polyhedral model **meets its Waterloo**)

Dynamic Resource Allocation

- Long running computationally intensive multi-core program (mixed memory/compute bound)
- Dynamic changes to operating environment
 - Reduction in cache resources
 - Change to number of available processors
 - Faults (in the future)
- Dynamically modify tile sizes
 - Allows adaptation to both changes
 - May need data remapping to improve locality

Details

Generate code (like checkpoint-restart) to

- Execute periodically
 - Evaluate a control decision to determine cost-benefit of changing tile sizes/remapping
 - Myopic decision may lead to regret
 - Probabilistic model of the arrival of dynamic changes to machine state
 - Model as a POMDP

Conclusion

- Polyhedral model for affine program analysis – we go beyond that to
 - Dynamic dependences
 - Non-polyhedral iteration spaces
- Optimal control using POMDPs provides a foundation for UAV control
 - Mathematically elegant and rigorous
 - Extensions to target ambiguity