Team Amalgam

Exact, Discrete, Multiobjective Optimization

Joseph Hong, Chris Kleynhans, Ming-Ho Yee, Atulan Zaman

Outline

Project and Customer

Development Practices

Current Progress

Next Steps

Summary

Project and Customer

Project

Optimize **Moolloy**, an implementation of the *Guided Improvement Algorithm* (GIA) for solving multiobjective optimization problems

Customer

Professor Derek Rayside

Development Practices

Source Control: Git hosted on GitHub

Build System: Ant and Waf

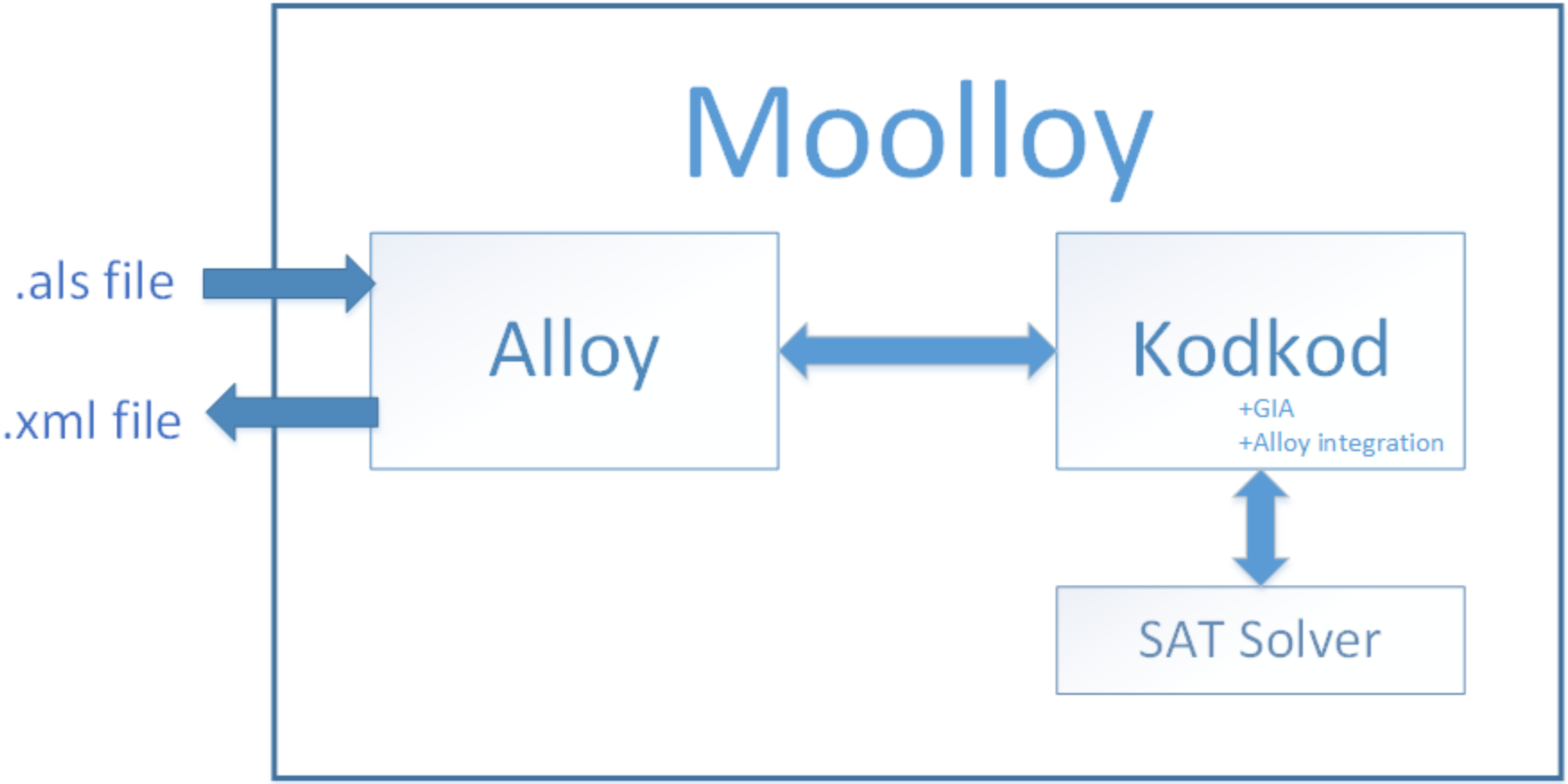
Build Schedule: Continuous integration with Travis

Regression Tests: JUnit on Travis + Dashboard

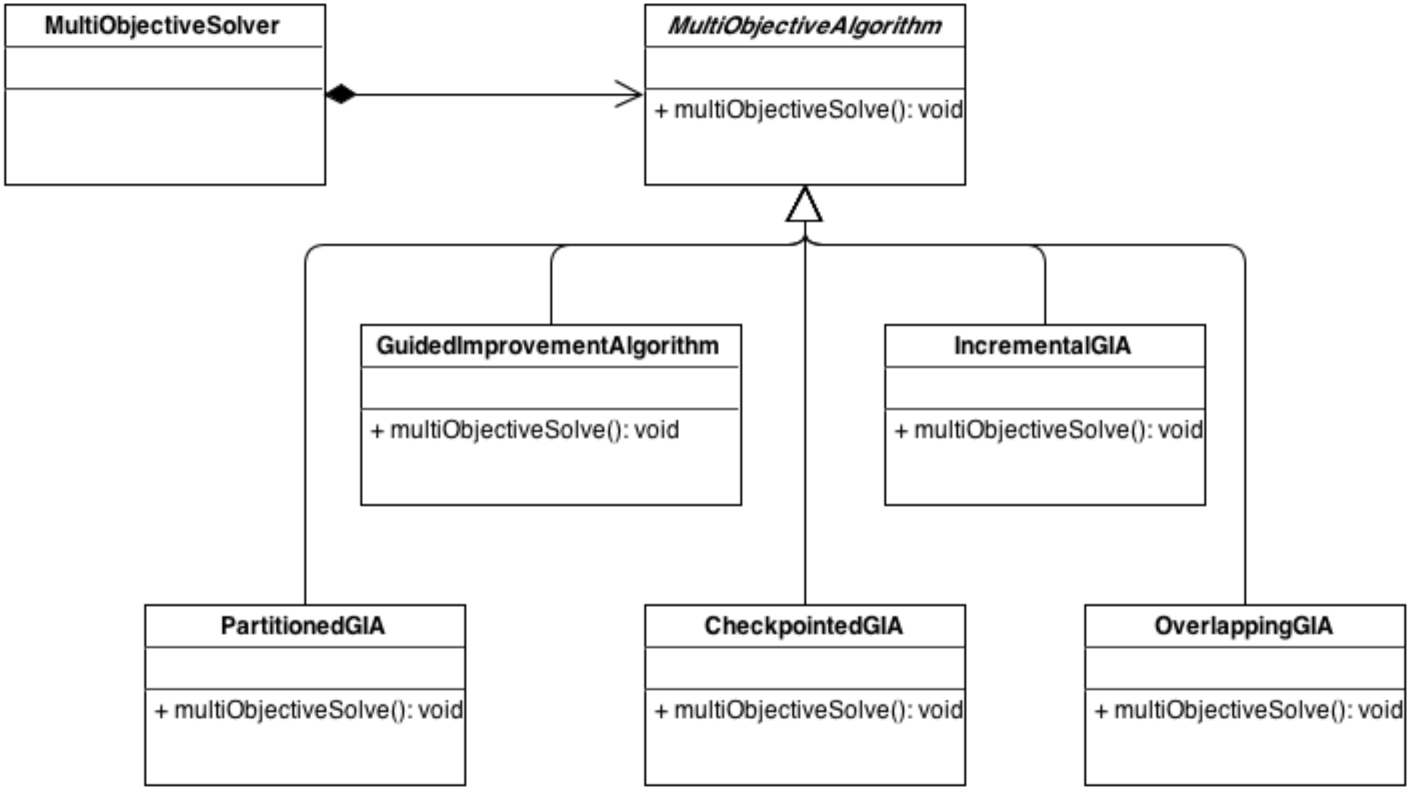
Bug DB+Schedule: GitHub Issues + Trello

Hallway Usability Testing: N/A

Current Status



Current Status


MultiObjectiveSolver
MultiObjectiveAlgorithm
+ multiObjectiveSolve(): void
GuidedImprovementAlgorithm
+ multiObjectiveSolve(): void
IncrementalGIA
+ multiObjectiveSolve(): void
PartitionedGIA
+ multiObjectiveSolve(): void
CheckpointedGIA
+ multiObjectiveSolve(): void
OverlappingGIA
+ multiObjectiveSolve(): void

Current Status
CheckpointedGIA

Removing constraints resets solver

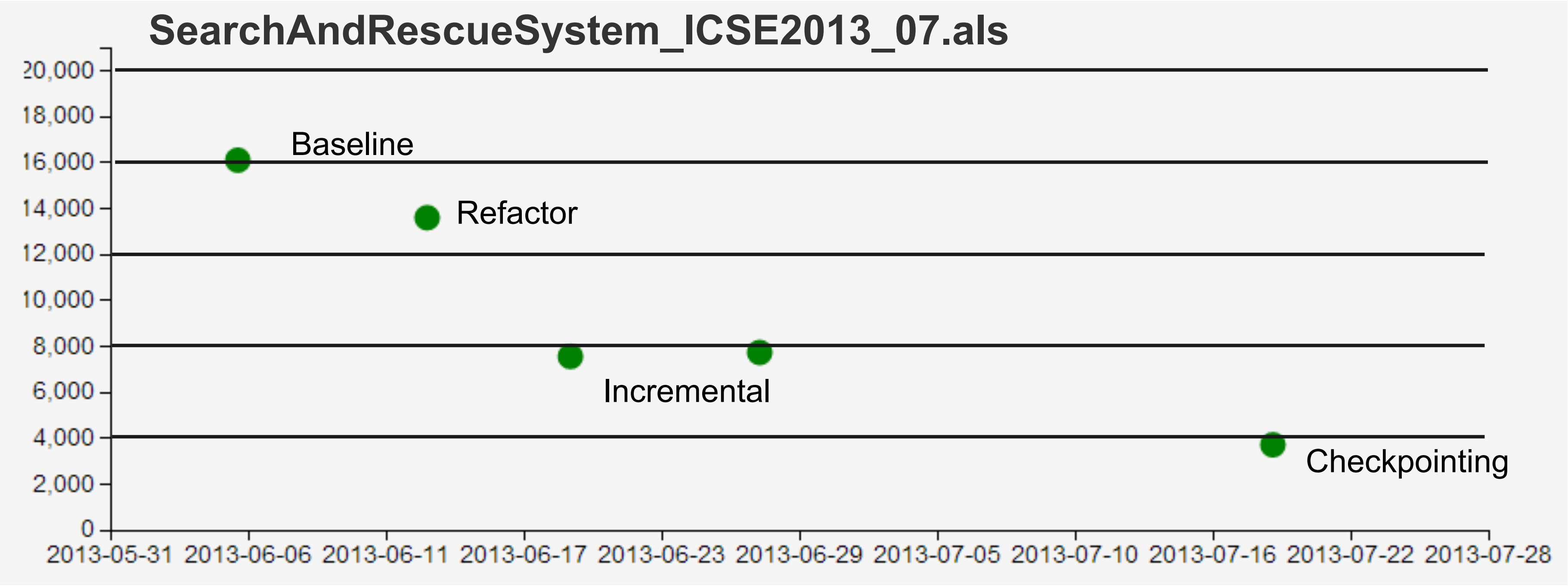
Idea: Add checkpointing

- Initial Implementation: Z3
  - Z3 already supports checkpointing
  - Added checkpointing to KodKod infrastructure
  - Default tactics are too slow for large SAT problems

Current Status

CheckpointedGIA

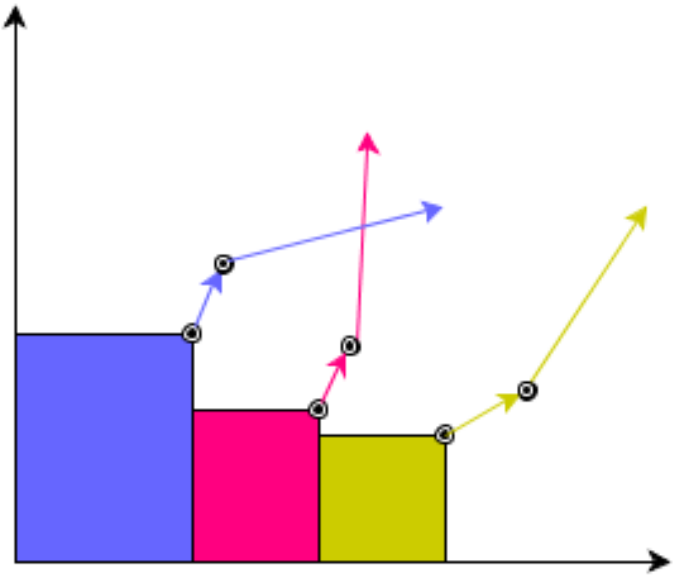
- Second Implementation: MiniSAT
  - Added checkpointing to MiniSAT



Current Status
OverlappingGIA

Idea: Parallelize the task so that different threads can search for different Pareto points

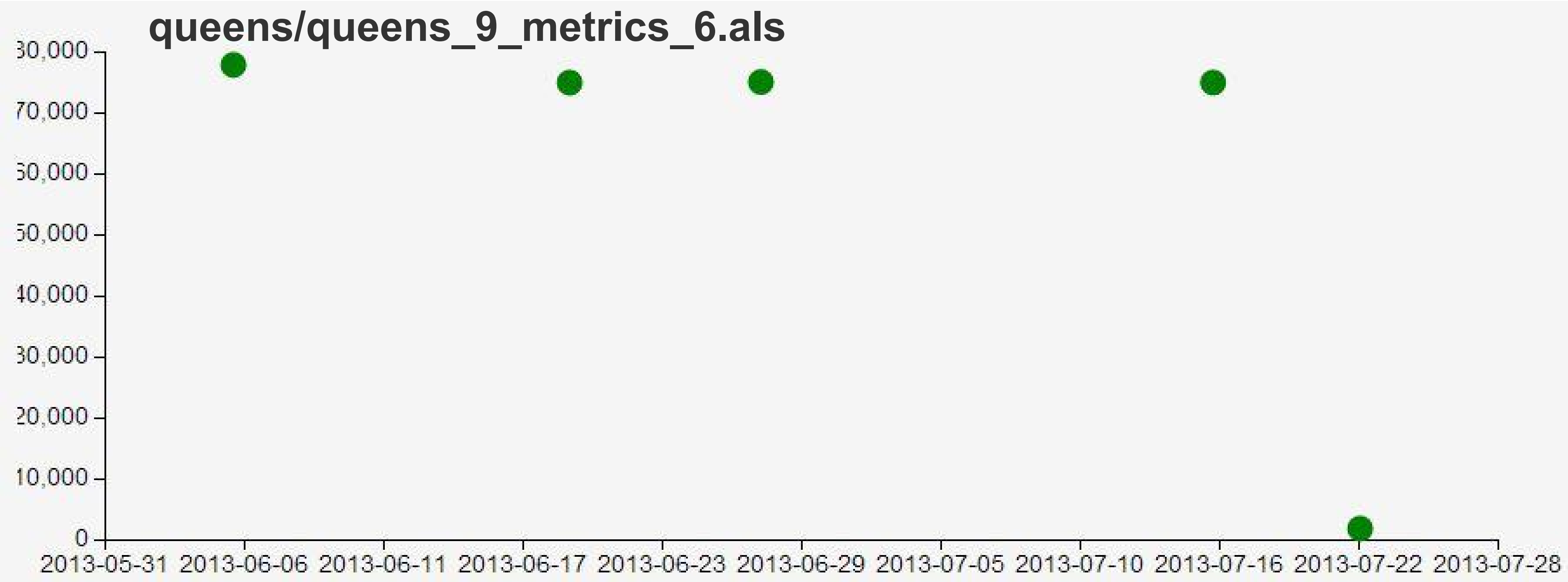
- Attempt to minimize duplicate work by generating unique starting points and deduplicating solutions



Current Status

OverlappingGIA

- Toy benchmark: 20 hours -> 30 minutes
- Case studies: mixed results



Current Status

PartitionedGIA

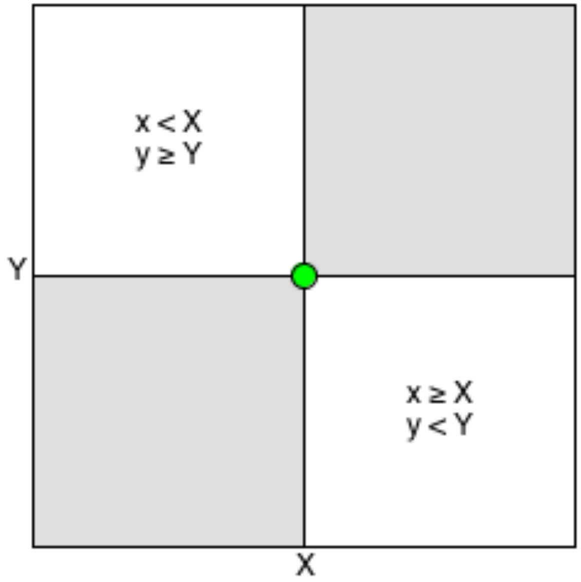
Idea: Split the search space into independent partitions

- Guarantee that locally optimal Pareto points are globally optimal
- Package partitions as tasks, and submit tasks to a thread pool

Current Status
PartitionedGIA

- Steps:
  - Find a single Pareto point
  - Split the search space based on that Pareto point
  - Run algorithm in each of the partitions, in parallel

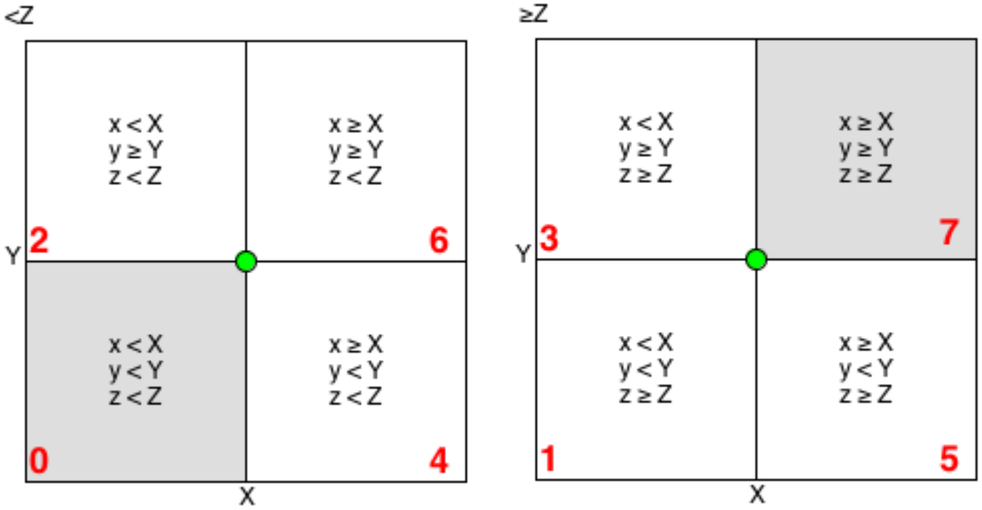
- Two objectives is a special case
  - Much more complicated in higher dimensions!

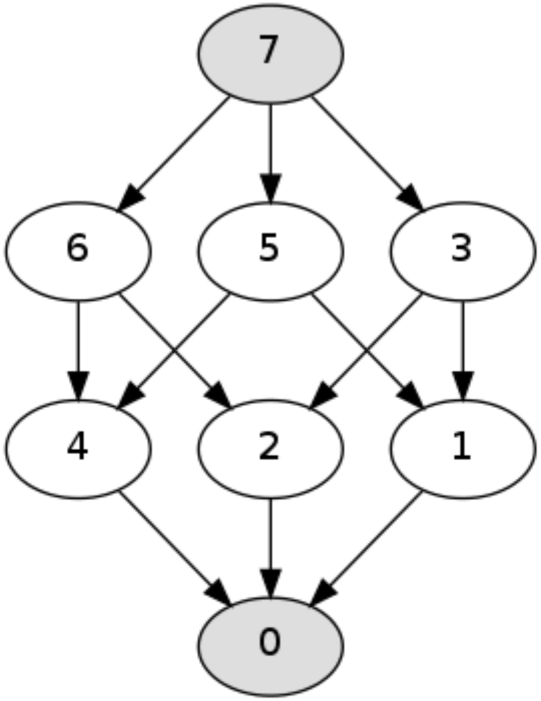


Current Status

PartitionedGIA

- This is trickier in higher dimensions
- Need to search in the partitions in a specific order





Current Status
PartitionedGIA

- Implementation is still going through code review
- Still running through our performance tests
- Preliminary results are mixed
  - Sometimes better than OverlappingGIA, sometimes worse

Next Steps

- Finish reviewing and finalizing implementations
- Implement Z3 as an SMT solver
- Improving or combining the implementations
- Validate ideas
    - Find case studies and run experiments

Summary

- Implemented three different ideas
  - PartitionedGIA, CheckpointedGIA, OverlappingGIA
- Results are very promising
- Still have more ideas to try out
- Want to validate ideas by running experiments on case studies