RESOURCE MANAGEMENT FOR HETEROGENEOUS COMPUTING SYSTEMS: UTILITY MAXIMIZATION, ENERGY-AWARE SCHEDULING, AND MULTI-OBJECTIVE OPTIMIZATION

Ryan Friese

Ph.D. Preliminary Examination

02/27/2014

Outline:

- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

Outline

- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

B. Khemka, R. Friese, L. D. Briceno, H. J. Siegel, A. A. Maciejewski, G. A. Koenig, C. Groer, G. Okonski, M. Hilton, R. Rambharos, and S. Poole, "Utility functions and resource management in an oversubscribed heterogeneous computing environment," *submitted to IEEE Transactions on Computers*, Oct. 2013

Heterogeneous Parallel Computing System

- interconnected set of different types of machines with varied computational capabilities

- workload of tasks with different computational requirements
- each task may perform differently on each machine
  - furthermore: machine A can be better than machine B for task 1 but not for task 2

- **resource allocation**:
  assign (map) tasks to machines
  to optimize some performance measure
  - NP-complete (cannot find optimal in reasonable time)
  - ex.: 5 machines and 30 tasks → 5^{30} possible assignments
    - 5^{30} nanoseconds > 1,000 years!
  - use heuristics to find near optimal allocation

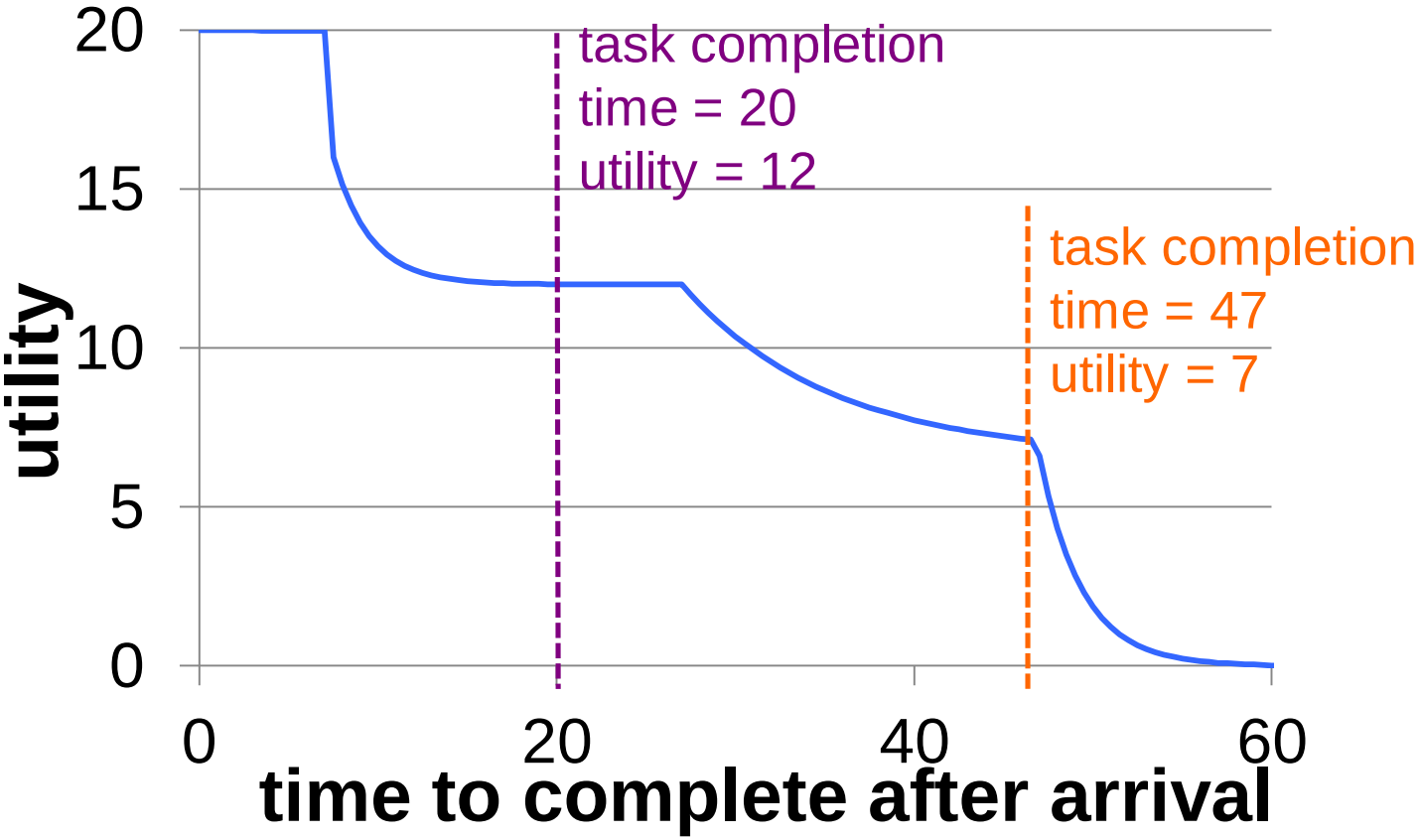
Colorado State University

Performance of Heterogeneous Systems

- how is the performance of a scheduler measured for a parallel or distributed computing system?
  - **homogeneous** case
    - utilization (not a good metric)
  - **oversubscribed heterogeneous** case
    - oversubscribed: total work exceeds capacity of system
    - heterogeneous: machines perform differently for a task
    - high utilization could mean bad performance because of a bad resource allocation!
      - mapping task to **worst** machine gives **high** utilization
    - measure the amount of useful work accomplished
      - use a flexible **utility function** model
    - based on a compute facility and workload being investigated by the Extreme Scale Systems Center at Oak Ridge National Lab

Measuring System Performance

- measured over a day with tasks arriving dynamically
- each task has a utility function (priority, urgency, utility class)
- for each task, consider the utility value it generated
  - value of utility function for the task when it completed
- **goal:** maximize sum of these values over all tasks



Colorado State University

Contributions

- model to create utility functions using the three parameters: priority, urgency, utility class
- model of the given DOE/DoD heterogeneous computing system and its intended workload
  - special-purpose machine that execute certain special-purpose tasks faster
  - sinusoidal and bursty arrival patterns for general and special-purpose tasks respectively
- design of a performance metric for schedulers in oversubscribed heterogeneous computing systems
- design and analysis of twelve heuristics to perform scheduling
- exploration of heuristic variations (including the dropping of low utility earning tasks)

Outline

- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

B. Khemka, R. Friese, S. Pasricha, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, R. Rambharos, and S. Poole, "Utility Driven Dynamic Resource Management in an Oversubscribed Energy-Constrained Heterogeneous System," *submitted to Sustainable Computing Journal (SUSCOM) Special Issue on Energy Aware Resource Management and Scheduling (EARMS)*, Mar. 2014

Motivation: Energy-Awareness

- energy consumption on high performance computing systems – very expensive
- in general there is a correlation between increasing compute power and increasing energy consumption
- how can we maintain or increase performance while using less energy?
  - energy-aware resource allocation

faster performance

greater energy requirements

billions spent in electricity costs

Colorado State University

Environment Description

- have an annual energy constraint and an accordingly scaled 24 hour energy constraint
- we simulate two days
  - first day is to bring system up to steady-state (day *i*–1)
  - performance is evaluated for second day (day *i*)
- heuristics are not allowed to permanently drop a task based on energy consumption
  - can postpone tasks consideration to next mapping event
- batch-mode heuristics are used to make allocation decisions
- utility earned and energy consumed by a given day are scaled based on the portion of task's execution on that day

Contributions

- designed and analyze four new batch resource management heuristics that try to maximize utility given an energy constraint
  - compared these against three previous batch heuristics
- designed a custom energy filtering mechanism
  - adapts to energy remaining in the day
  - enforces “fairness” in energy consumption
  - makes utility-aware heuristics become energy-aware
- sensitivity analysis on the level of energy-awareness for all heuristics
- method to create low TMA and high TMA environments based on a reference environment (without changing other heterogeneity measures)
  - analysis of all heuristics in these environments

Outline

- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

R. Friese, B. Khemka, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, J. Rambharos, G. Okonski, and S. W. Poole, "An analysis framework for investigating the trade-offs between system performance and energy consumption in a heterogeneous computing environments," in *22nd Heterogeneity in Computing Workshop (**HCW 2013**), in the proceedings of the 27th International Parallel and Distributed Processing Symposium (IPDPS 2013)*, May 2013.

Bi-Objective Optimization for Energy-Aware Scheduling

- why is it important to study bi-objective optimization?
  - real world problems often have multiple objectives
  - objectives may conflict with each other
- used to understand how different resource allocations trade-off between:
  - utility earned
  - energy consumed
- there is not a single metric to represent both objectives
- will be performed in a static post-mortem environment
- will lead to the design of dynamic scheduling heuristics considering utility and energy

Task and Machine Types

- task execution type – similar computational requirements
- machine execution type – similar performance capabilities
- we use an **Estimated Time to Compute (ETC)** matrix
  - gives estimated time for executing each task type on each machine type
- we use an **Average Power Consumption (APC)** matrix
  - gives power consumed for executing each task type on each machine type
- in real world: use historical data, experiments, benchmarks
- simulator uses a synthetic workload extrapolated from real data
- **special** (limited) **purpose** machine types
  - execute some task types much faster than the other machines types
  - may be incapable of executing the other task types

Calculating Energy Consumption

- Estimate Time to Compute (ETC) - the execution time of a given task on a given machine
- Average Power Consumption (APC) - the average power consumption of a given task on a given machine

ETC values (seconds)

	M1	M2
T1	8	10
T2	9	12
T3	7	11

APC values (watts)

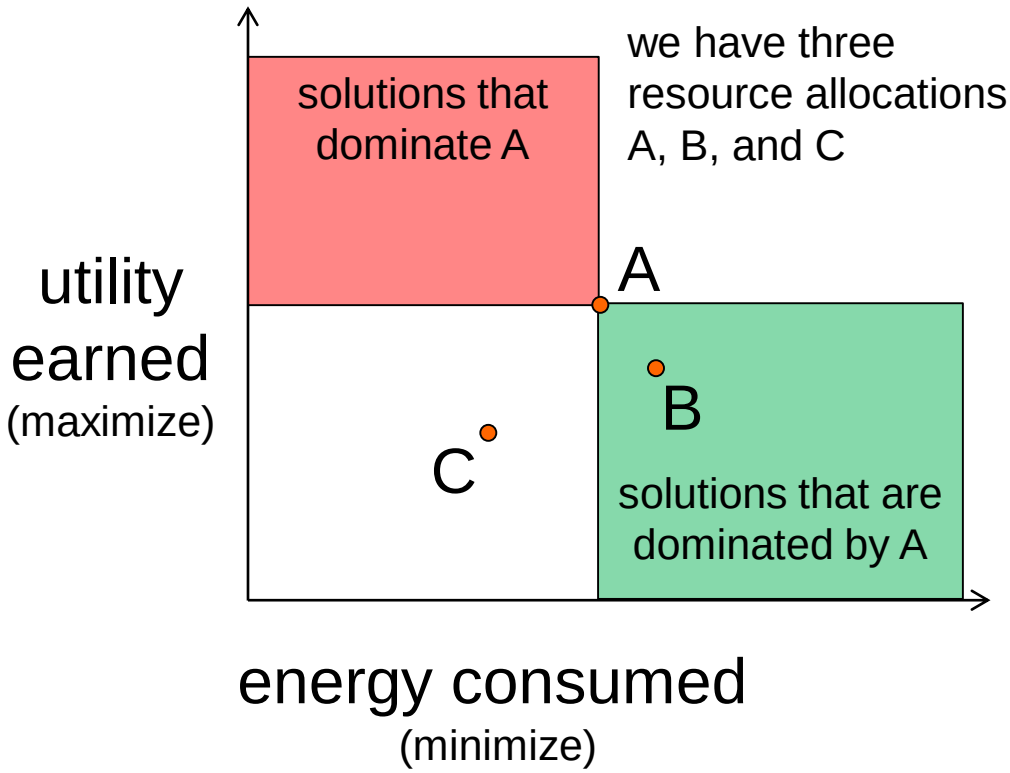
	M1	M2
T1	115	95
T2	105	87
T3	125	90

energy of T3 on M2 = 11 seconds * 90 watts = 990 joules

Assumed Environment

- **oversubscribed** environment
  - total work exceeds capacity of system
- tasks are assumed to be **independent**
  - communication is not required between the tasks and there are not any precedence constraints
- tasks are assumed to be **serial**
  - execute on a single machine
- static analysis is performed
  - of a dynamic system including task arrival times
- real world: to analyze a given system with an intended workload one could use a system trace of task arrivals

Pareto Fronts



- B is dominated by A because A uses less energy while earning more utility, thus A is the better solution
- neither A nor C dominate each other because A is better for utility but C is better for energy, neither is better in both

- a Pareto front contains all the solutions which are not "dominated" by any other solution
- Pareto fronts facilitate trade-off analysis

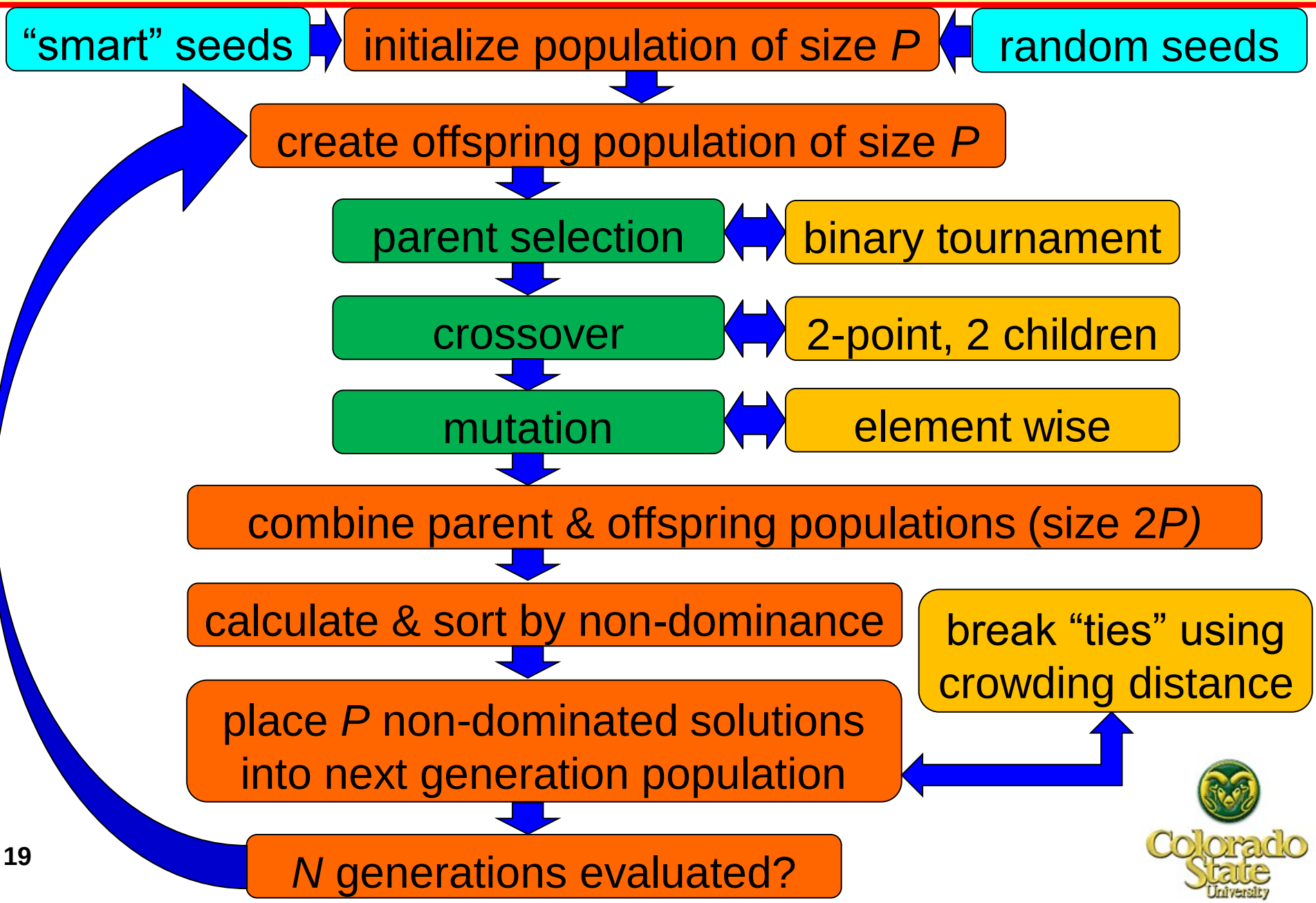
Solving Bi-Objective Problems with Genetic Algorithms

- genetic algorithms are search heuristics used to find solutions to optimization problems
- genetic algorithms try to mimic the process of natural evolution
- solutions “evolve” through time by passing on useful traits
- given enough time, the solutions will converge towards the set of optimal solutions
- preferably solutions should be diverse and evenly distributed across the Pareto front

Genetic Algorithm Setup

- **gene:** most basic unit, represents a task to machine mapping
  - machine on which the task executes
  - the global scheduling order of the task
- **chromosome:** consists of multiple genes and represents a complete resource allocation
- **population:** consists of multiple chromosomes and represents a set of solutions

NSGA-II Overview



Colorado State University

Environment Setup

- objective functions
  - maximize utility earned
  - minimize energy consumed
- all tasks are executed
- seed genetic algorithm with various heuristics

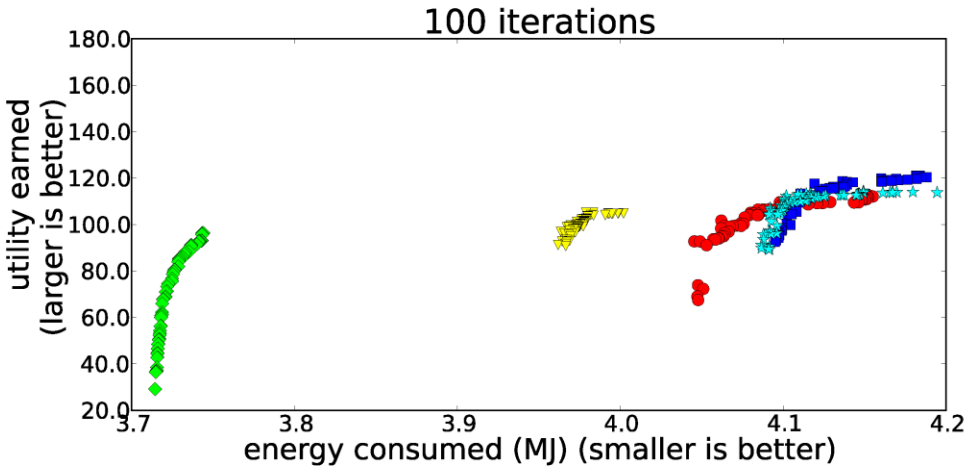
Seeding Heuristics

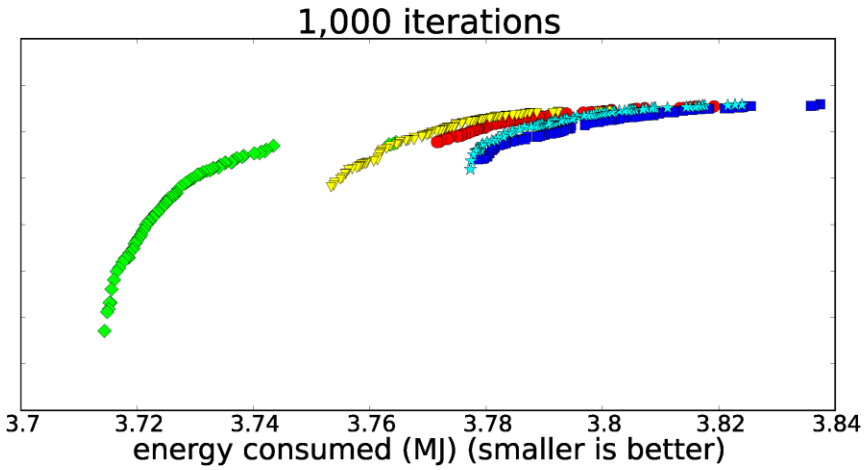
- Max Utility
  - map task to the machine that will earn the most utility
- Min Energy
  - map task to the machine that consumes the least energy (energy = execution time * power consumption)
- Max Utility-Per-Energy
  - map to the machine that will earn the most utility per unit of energy consumed
- Min-Min Completion Time
  - two-stage heuristic assigning tasks to machines, picking assignments with least completion time (completion time = task start time + execution time)

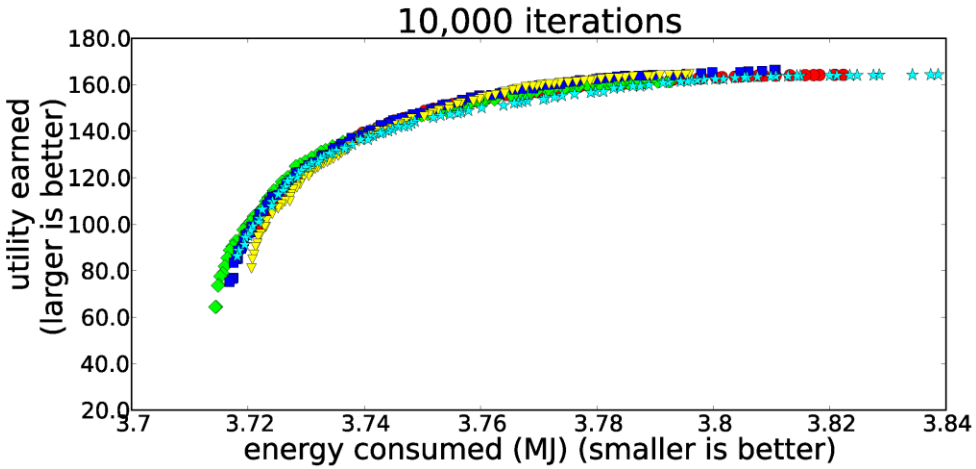
Small-scale Simulation

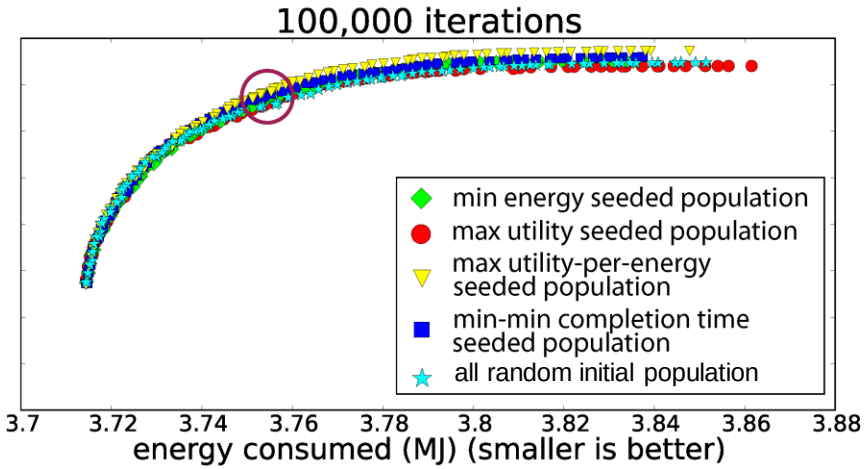
- real data for ETC and APC
- 9 general machines
  - 9 general machine types
- 250 tasks
  - 5 task types
- 100 chromosomes
- 100,000 iterations

Pareto Front Evolution for Real Data Set







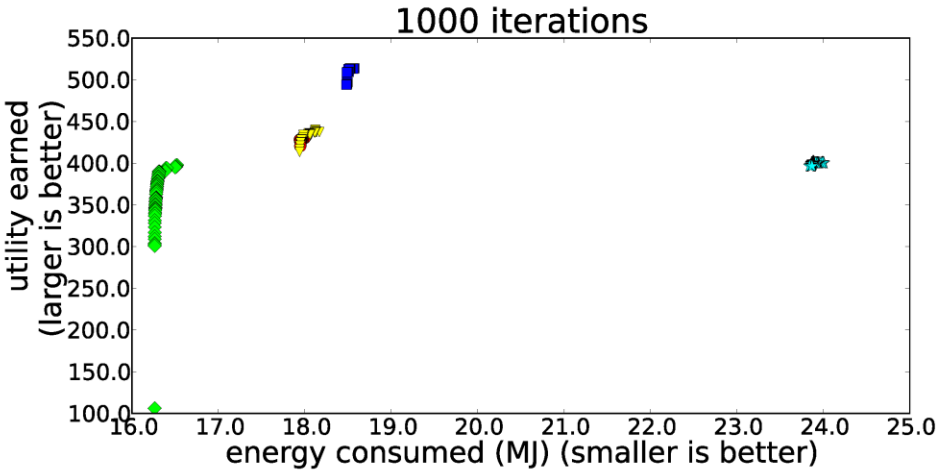


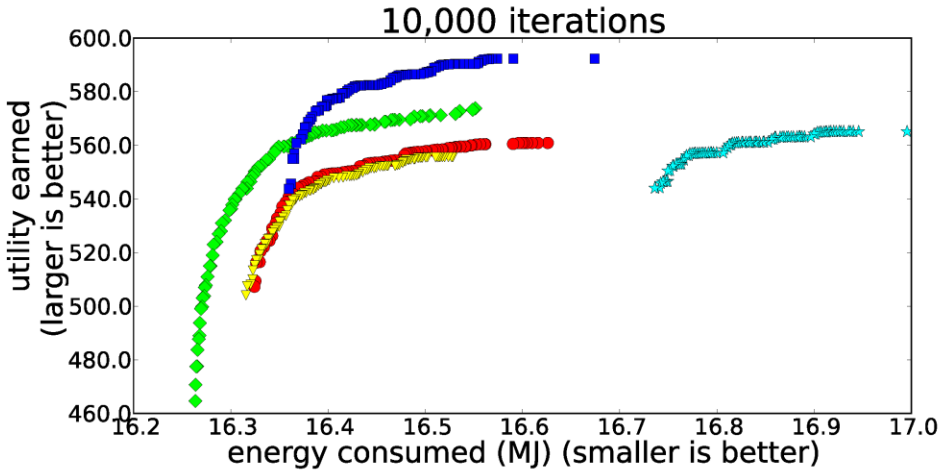
Colorado State University

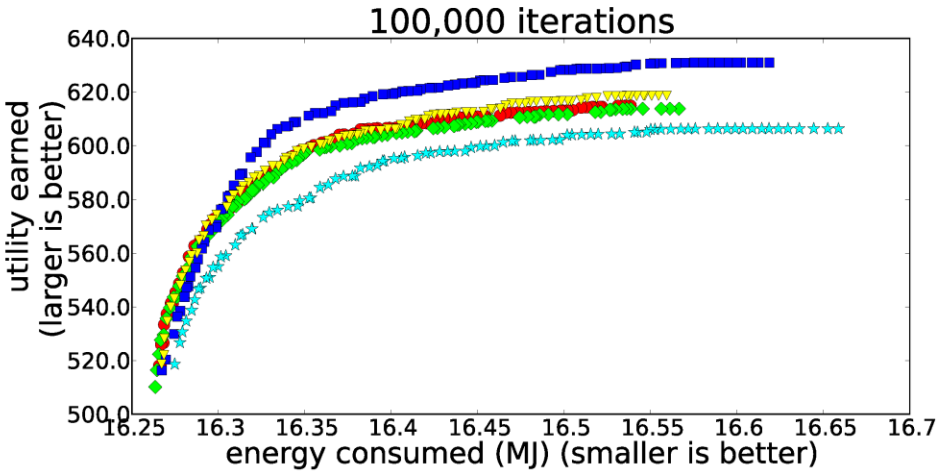
Large-scale Simulations – Synthetic Data

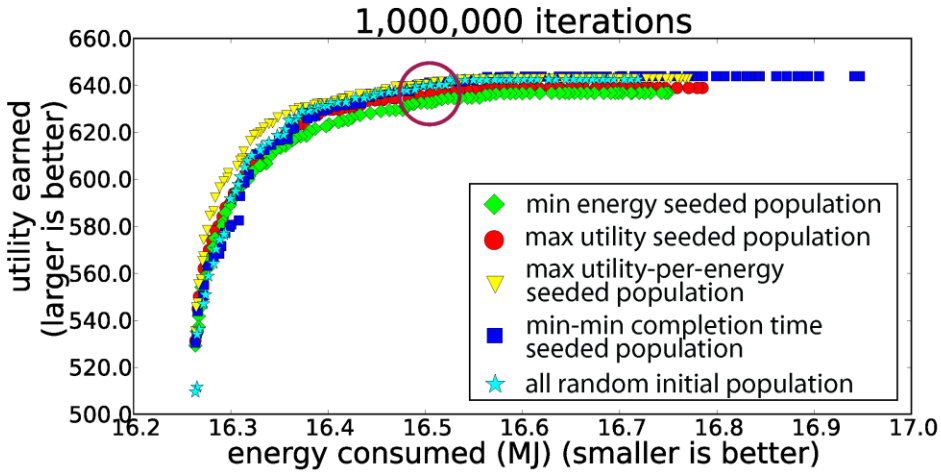
- 4 special machines
  - 4 special machine types
- 26 general machines
  - 9 general machine types
- ~1000 or ~4000 tasks
  - 30 task types (5 real, 25 synthetic)
- 100 chromosomes
- 1,000,000 iterations

Evolution for Synthetic Data Set with 1000 tasks



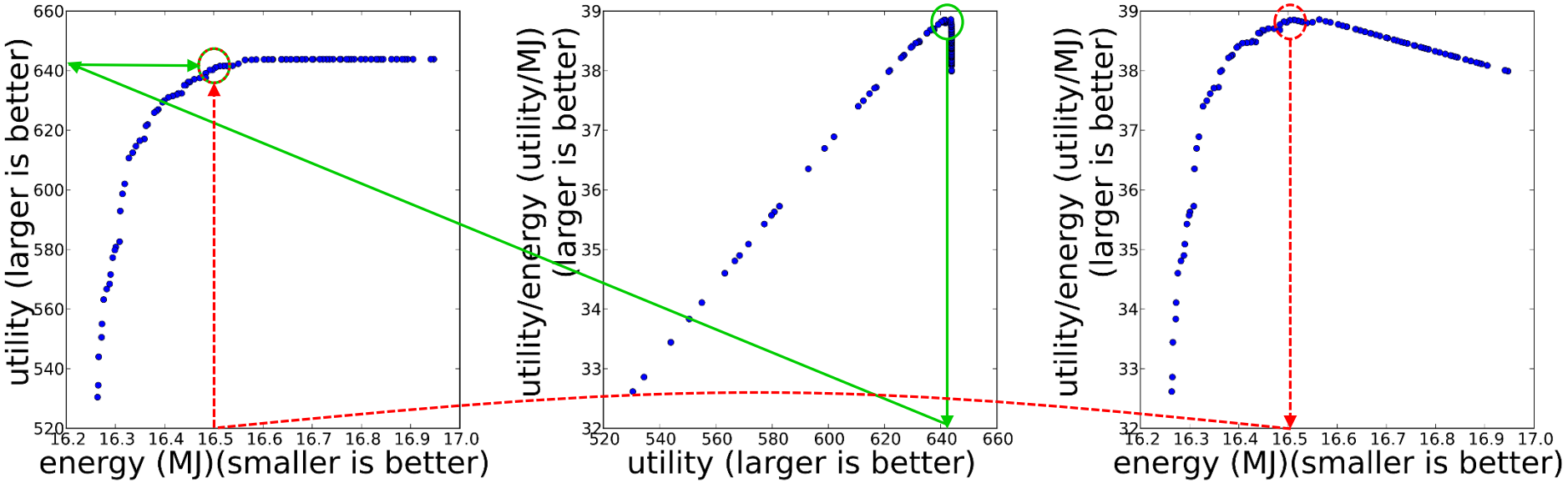






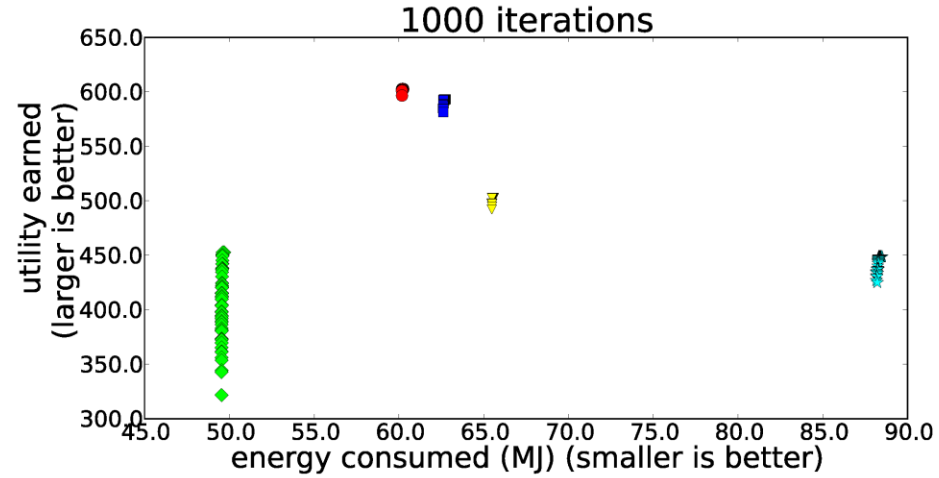
Colorado State University

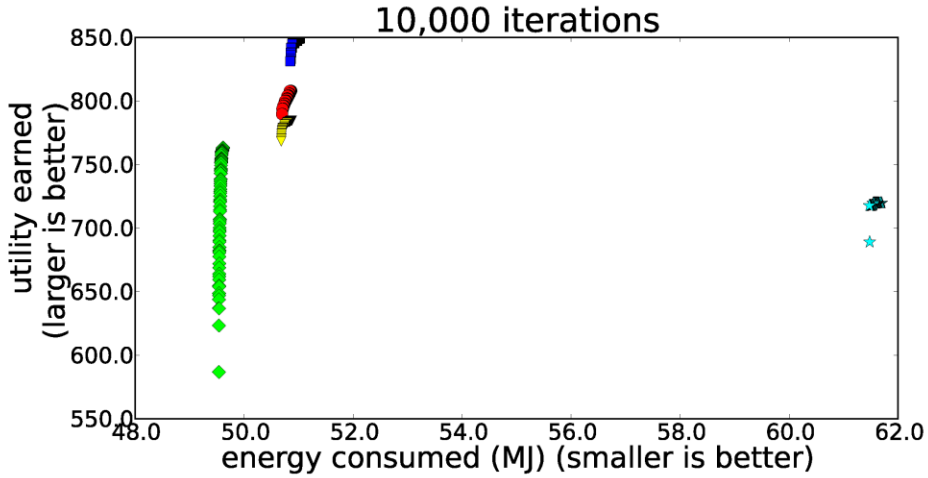
Understanding Performance Trade-offs

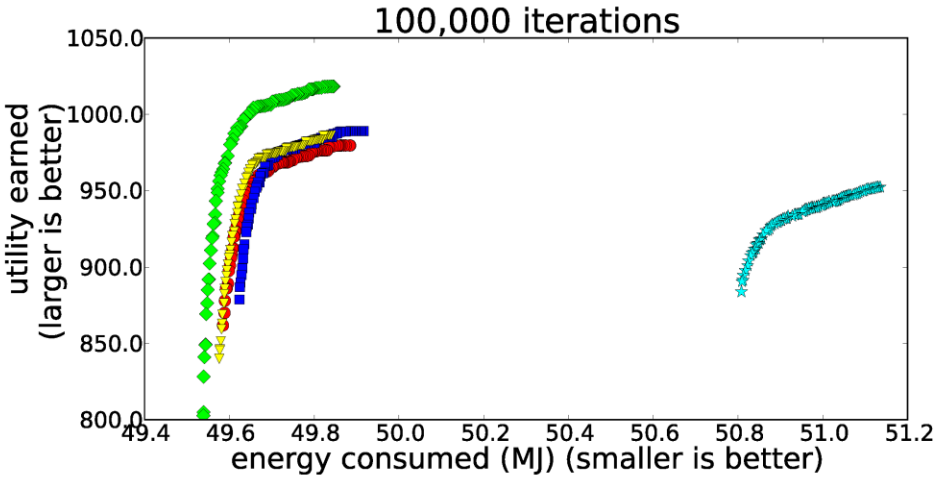


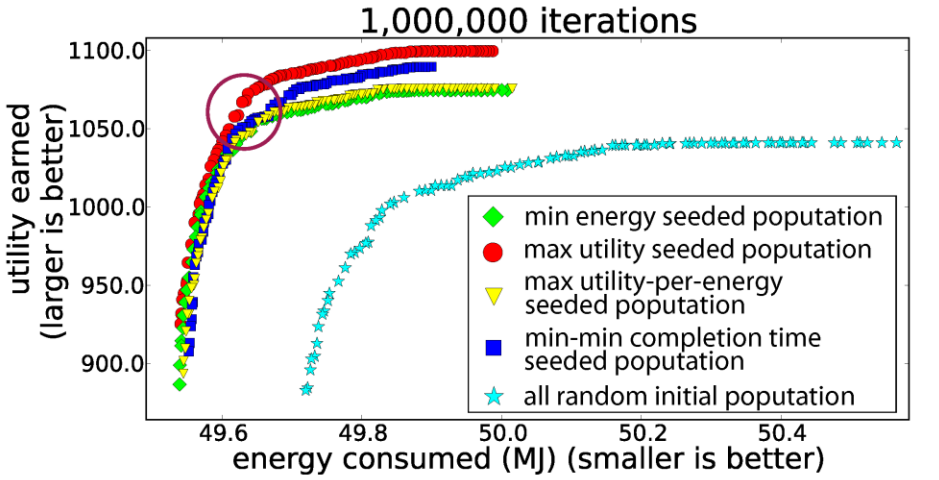
Colorado State University

Evolution for Synthetic Data Set with 4000 tasks









Colorado State University

Current Contributions

- adapted a popular multi-objective genetic algorithm for use with heterogeneous HPC resource allocation
  - chromosome structure
  - crossover and mutation operations
- modeled a bi-objective resource allocation problem
  - utility earned and energy consumed
  - based on compute facility and workload being investigated by the Extreme Scale Systems Center at Oak Ridge National Lab
- creating and evaluating many intelligent resource allocations
  - show how utility and energy change drastically in a system
- analyze the effects the different seeding heuristics have on performance of the algorithm

Future Work

- implement technique to drop low utility-earning tasks
- modeling P-states in our computing system
- stochastic GA variant with an objective to maximize robustness
- explore the island model (possibly with different seeds for the different islands) for the GA
- perform >2 objective optimization
  - utility vs. energy vs. dropped tasks
  - utility vs. energy vs. robustness

Outline

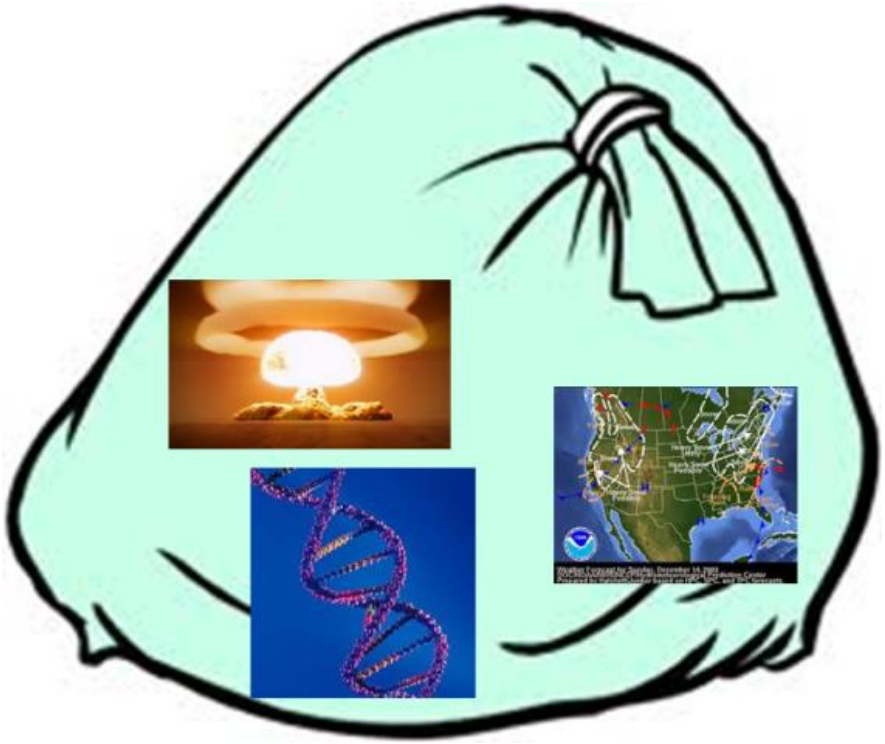
- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

R. Friese, T. Brinks, C. Oliver, A. A. Maciejewski, and H. J. Siegel, "Analyzing the Trade-offs Between Minimizing Makespan and Minimizing Energy Consumption in a Heterogeneous Resource Allocation Problem," in *The 2nd International Conference on Advanced Communications and Computation (INFOCOMP 2012)*, Oct. 2012.

R. Friese, T.Brinks, C. Oliver, A. A. Maciejewski, H. J. Siegel, and S. Pasricha, "A machine-by-machine analysis of a bi-objective resource allocation problem," in *The 2013 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2013)*, July 2013.

Environment

- static and offline environment
  - bag of tasks (batch)
  - every task in the workload is known *a priori*



Resource Allocations Objectives and Goal

- considers two objectives that typically conflict
  - **makespan**
    - total amount of time it takes for all the tasks in the batch to finish executing across all machines
  - **energy consumption**
    - total amount of energy consumed to execute all tasks within the batch

goal:

- evaluate trade-offs between makespan and energy consumption

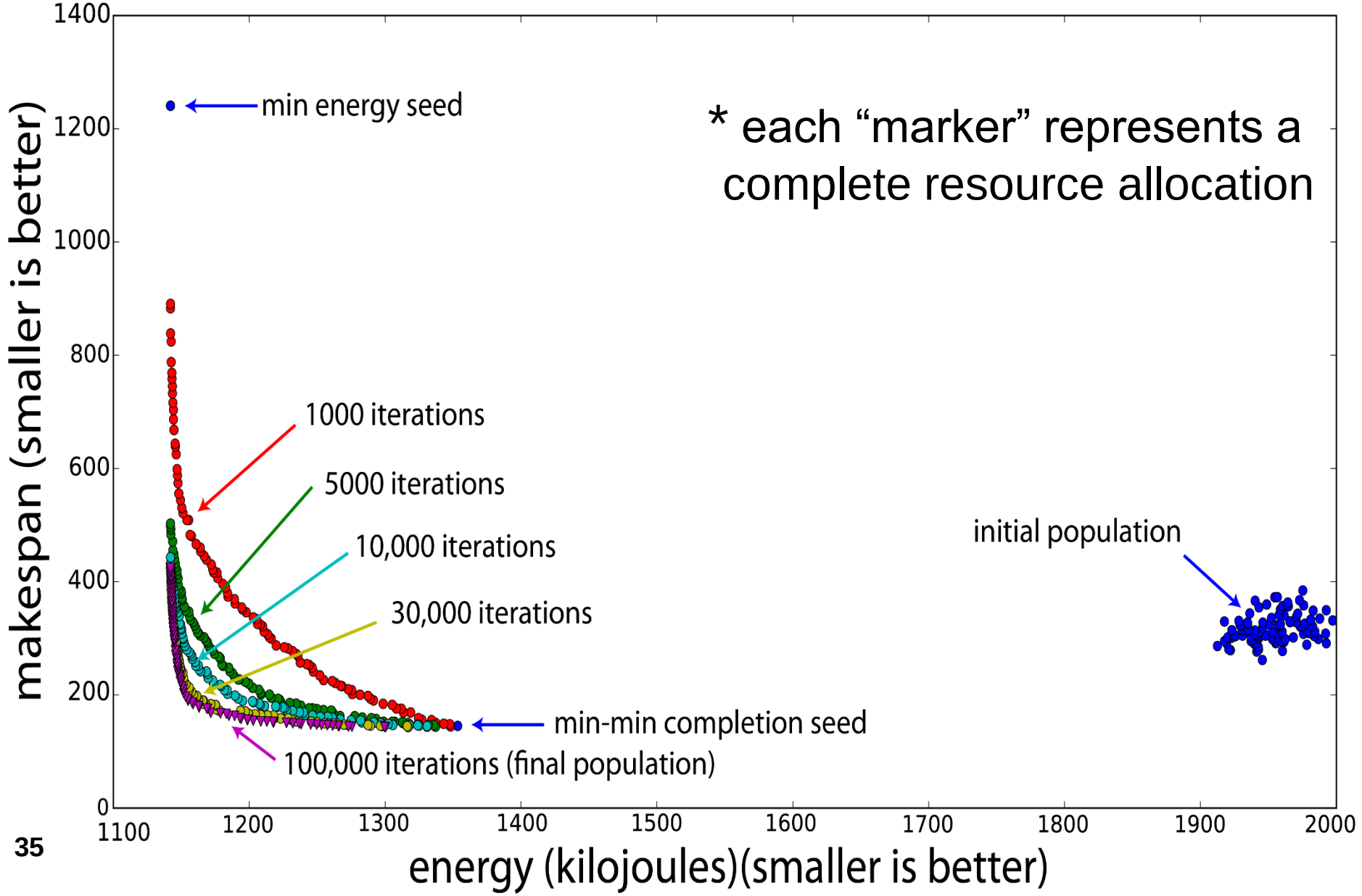
Seeding Heuristics for Initial Population

- one chromosome is the Min Energy seed
  - Min Energy
    - map task to machine that consumes the least energy (energy = execution time * power consumption)
- one chromosome is the Min-Min completion time seed
  - Min-Min Completion Time
    - two-stage heuristic assigning tasks to machines, picking assignments with least completion time (completion time = task start time + execution time)
- the rest of the chromosomes are created randomly

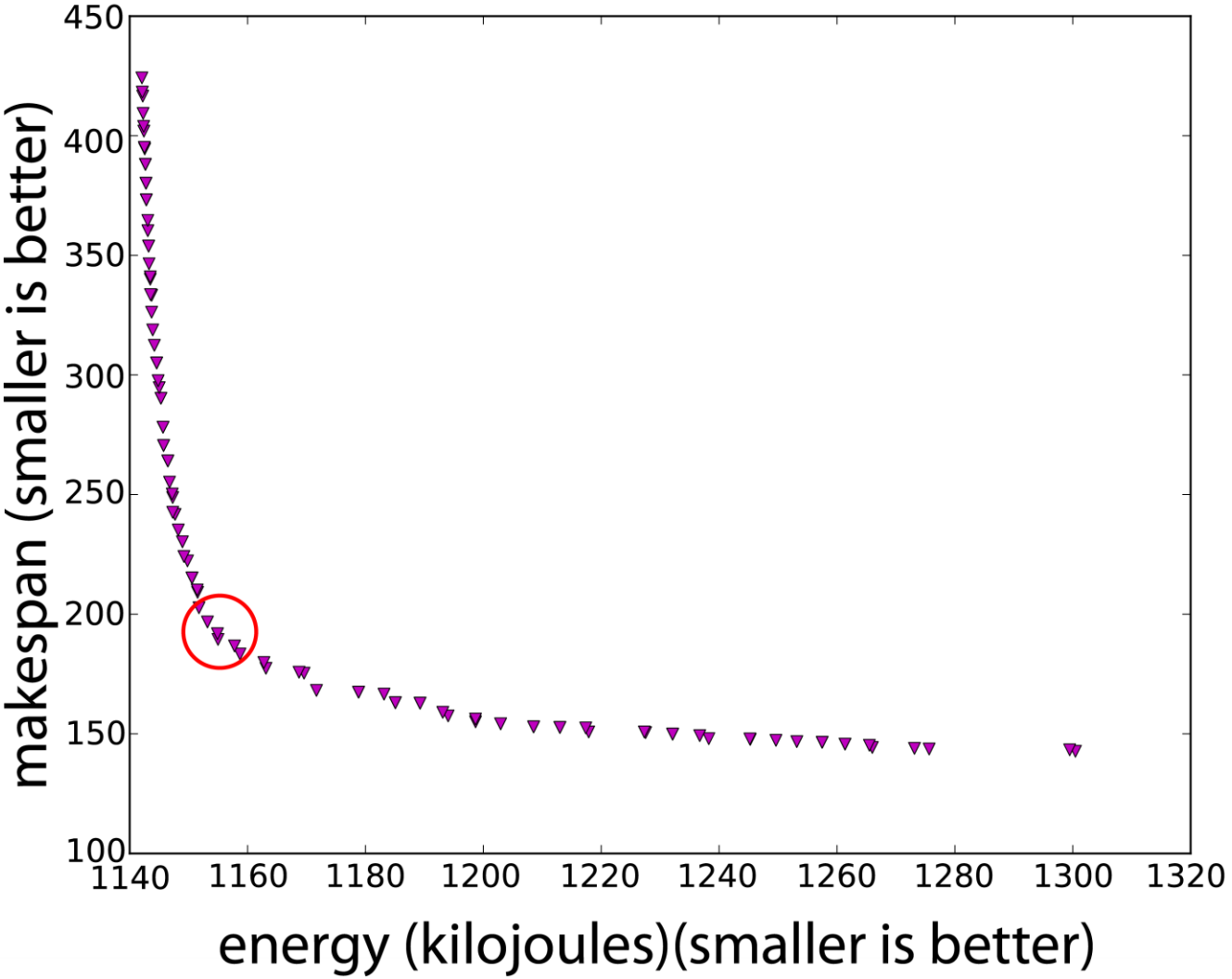
Simulation Setup

- 50 machines
  - 10 machine types
- 1000 task
  - 50 task types
- synthetically generated 10x50 ETC and APC matrices
- 100 chromosomes in genetic algorithm population
- 100,000 iterations of the genetic algorithm

Evolution of Pareto Front Through 100,000 iterations

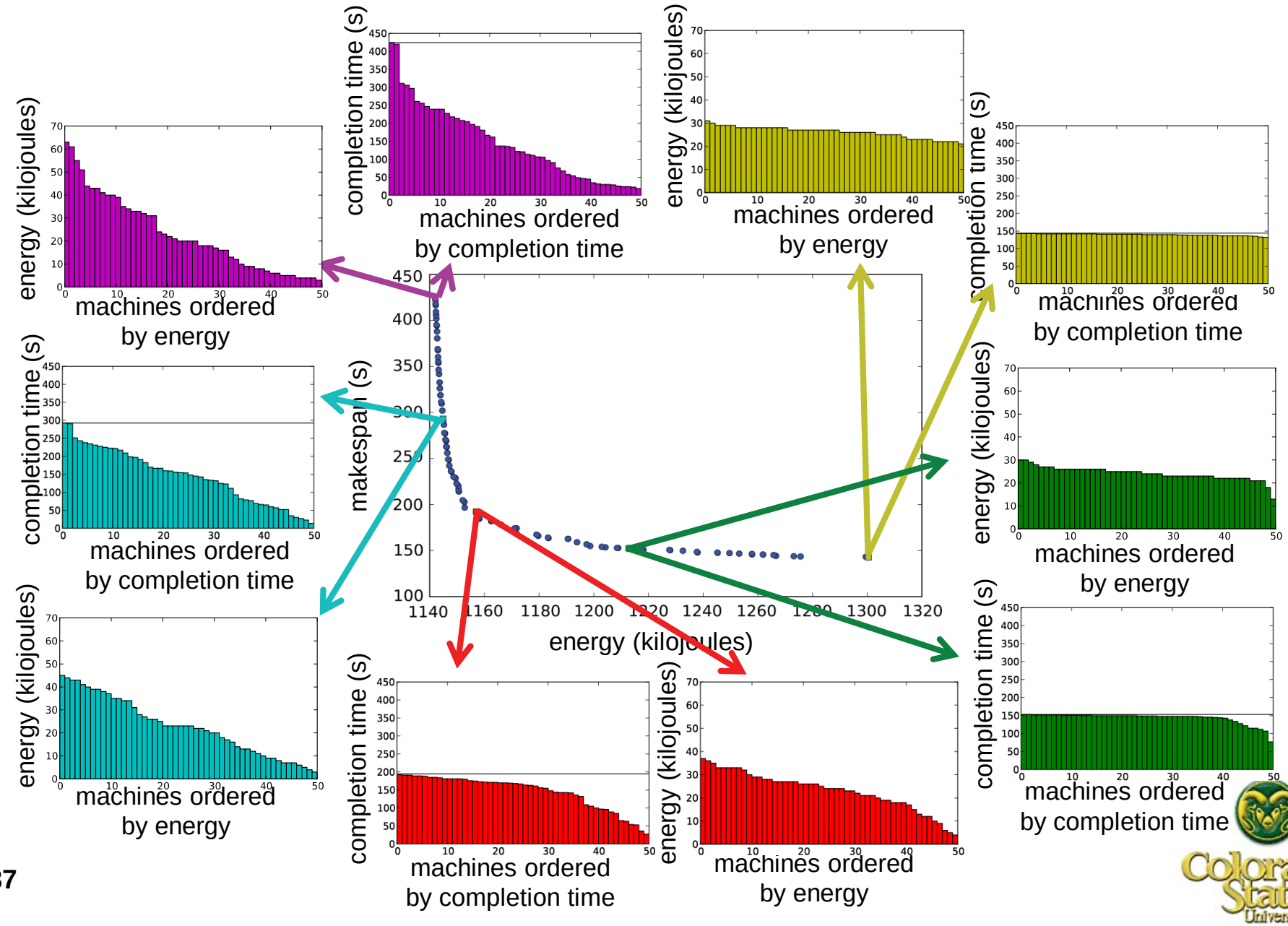


Final Pareto Front



Colorado State University

Analysis of Five Solutions from the Final Pareto Front

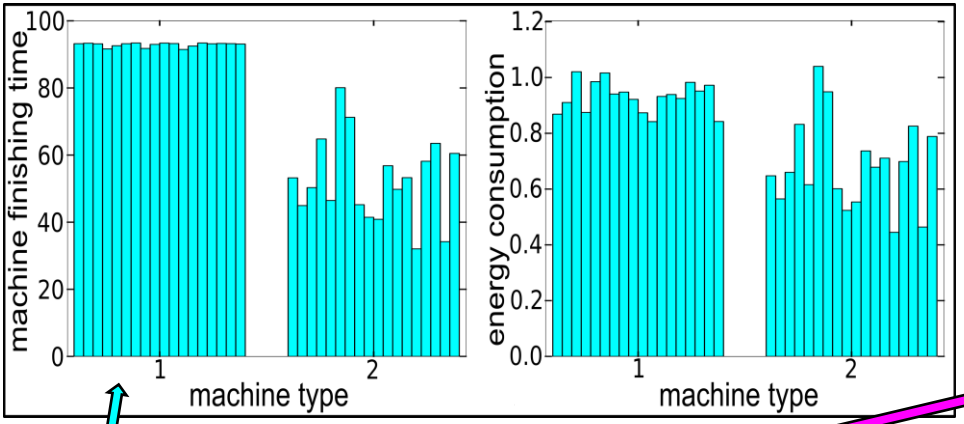


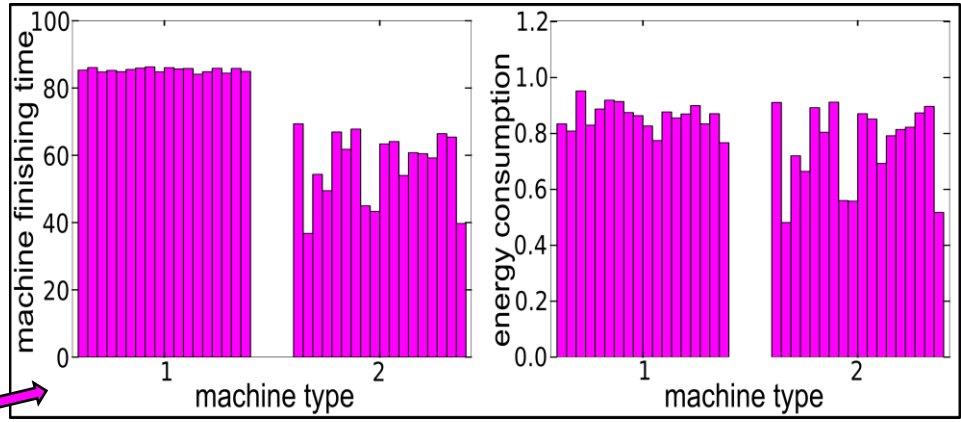
Colorado State University

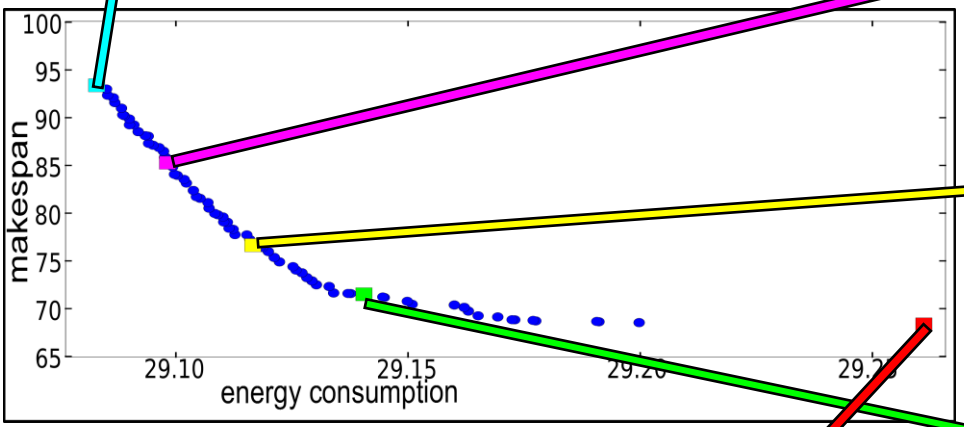
Simulation Setup

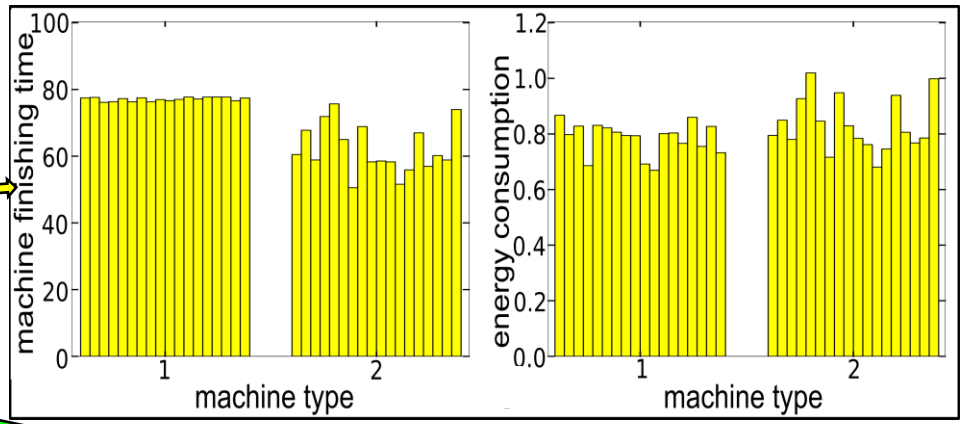
- to further understand the trade-offs between makespan and energy, and to illustrate the versatility of the approach three additional environments were modeled
- 36 machines
  - 2 machine types (18 machines per type)
  - 6 machine types (6 machines per type)
  - 9 machine types (4 machines per type)
- 9 machine types are based on real machines
  - the 2 and 6 machine types are subsets of the 9 machine types
- 1000 tasks
  - 30 tasks types
- Pareto fronts were generated using NSGA-II
- any algorithm that creates Pareto fronts could be used as well

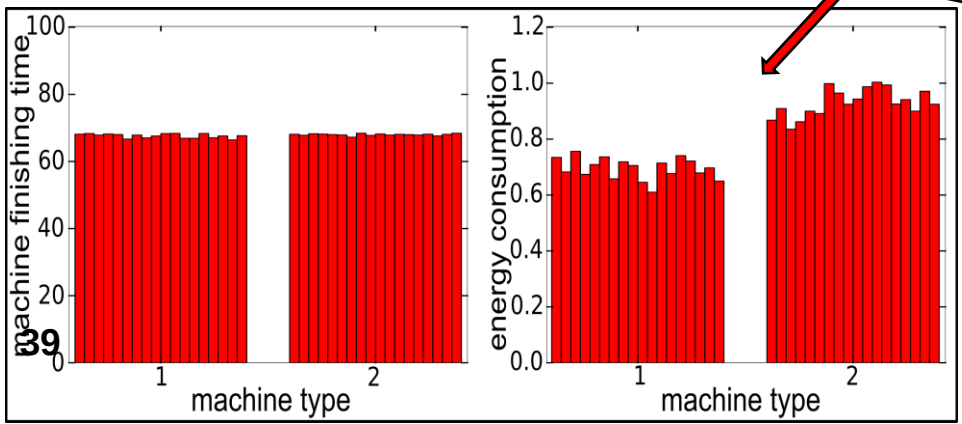
2 Machine Type Environment Analysis

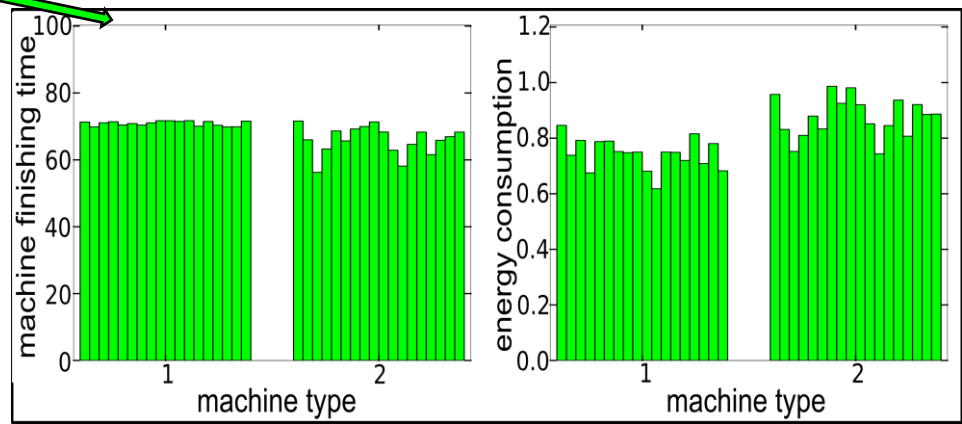




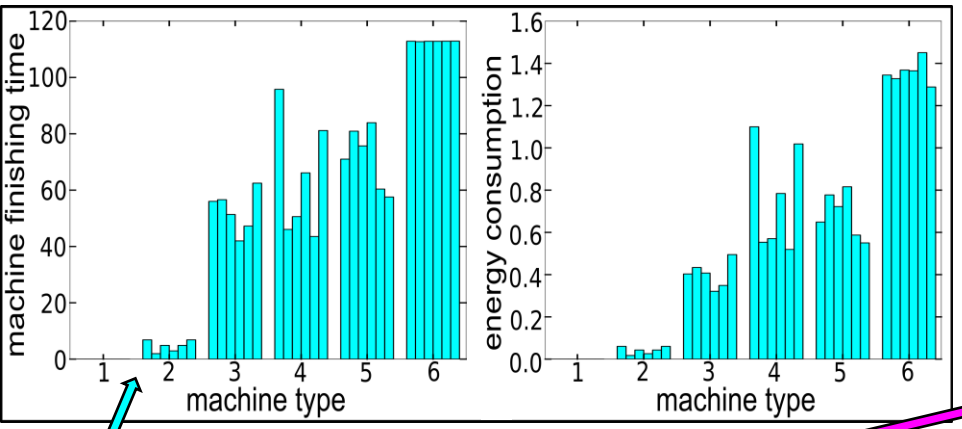


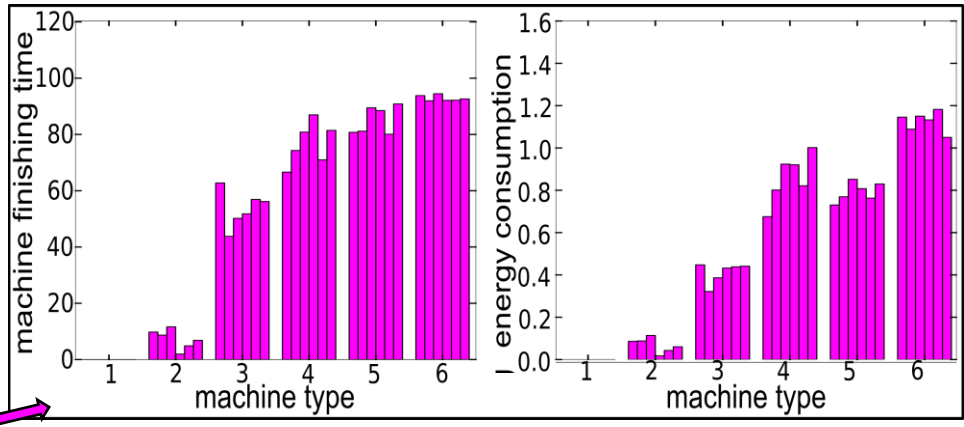


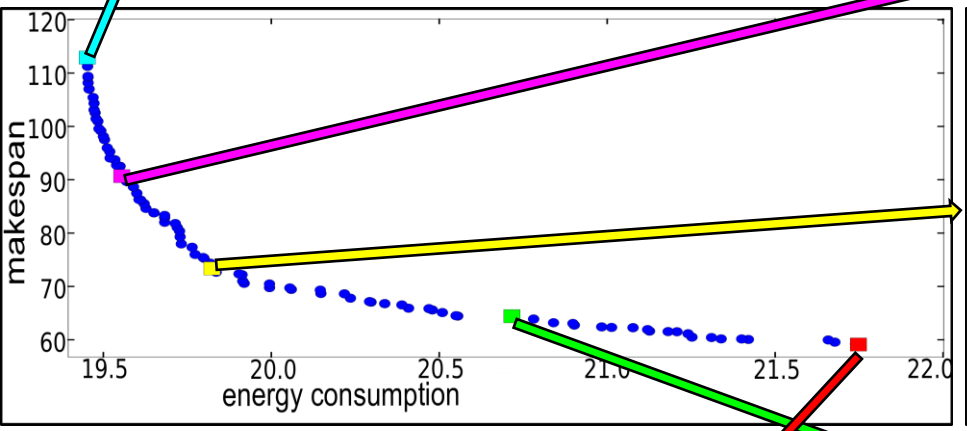


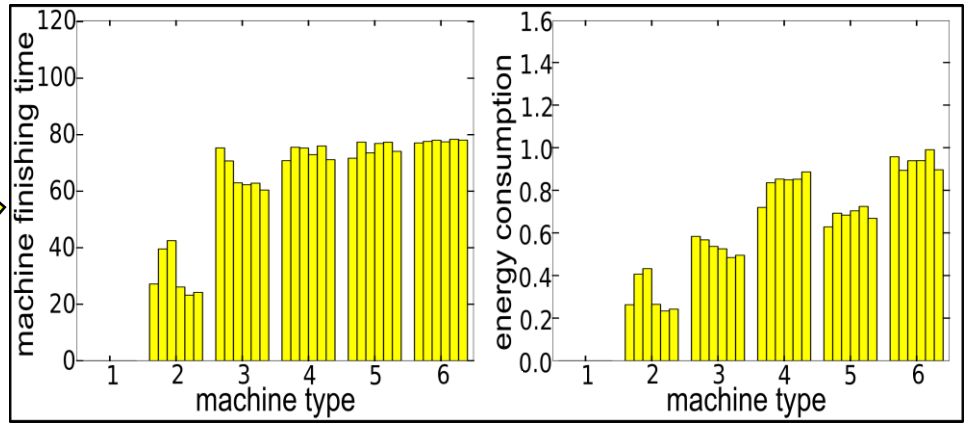


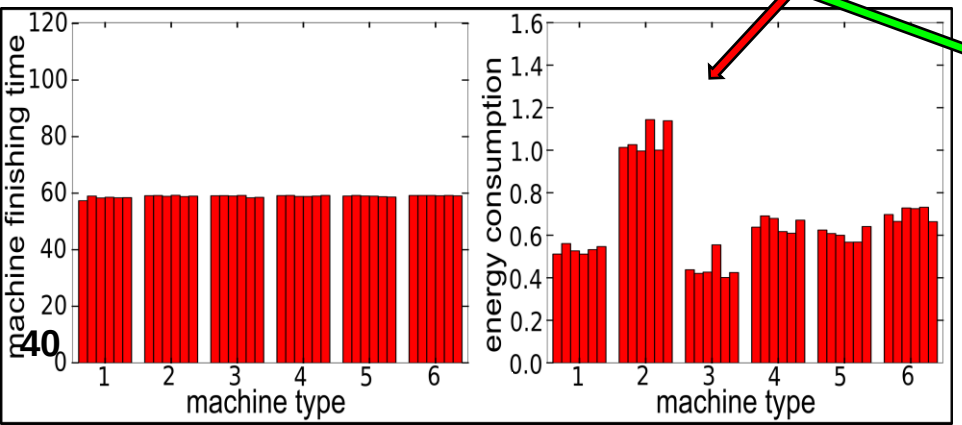
6 Machine Type Environment Analysis

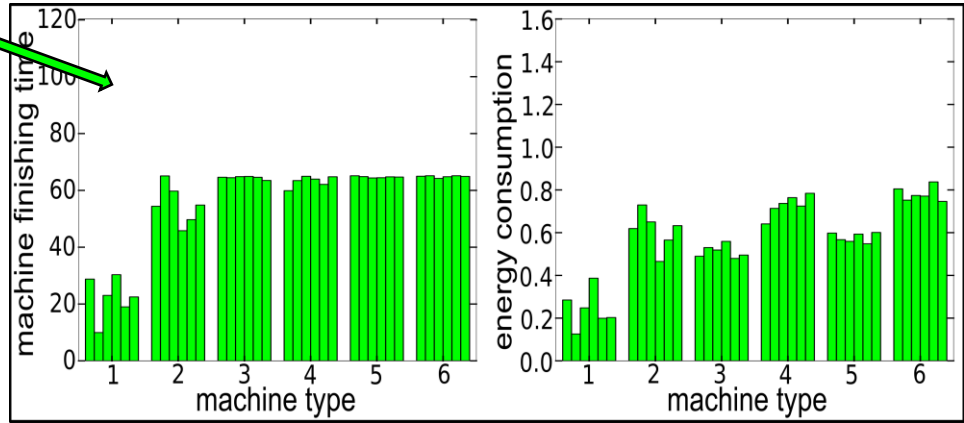




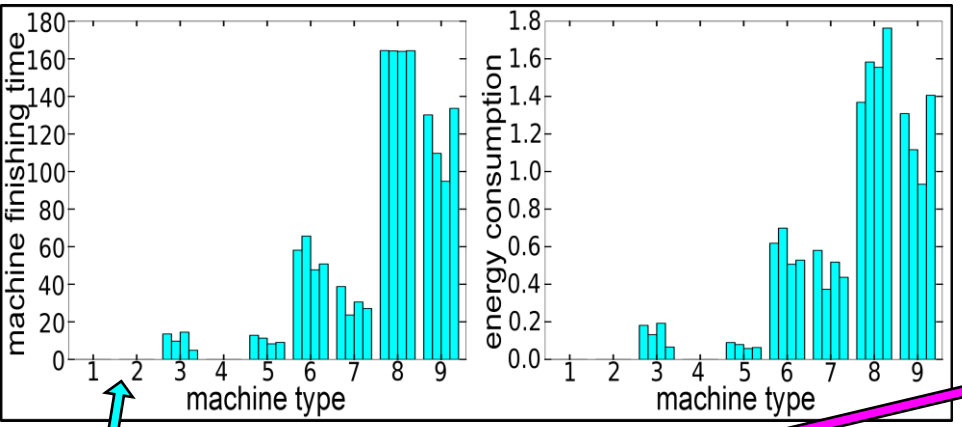


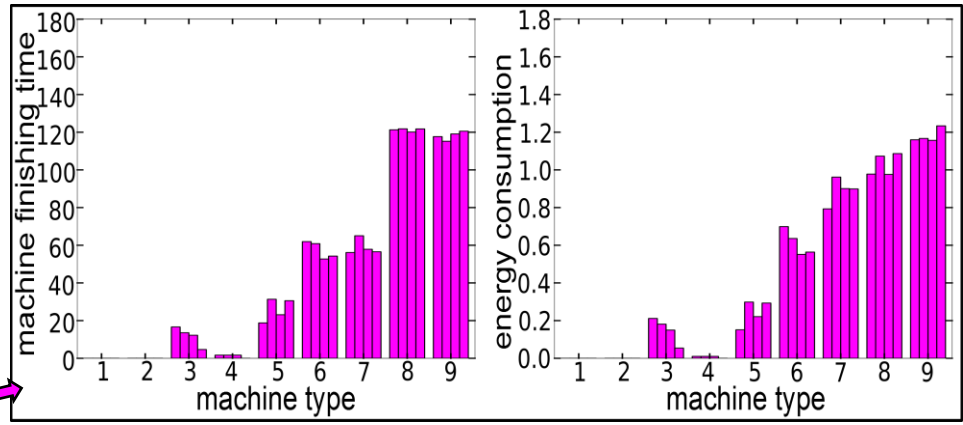


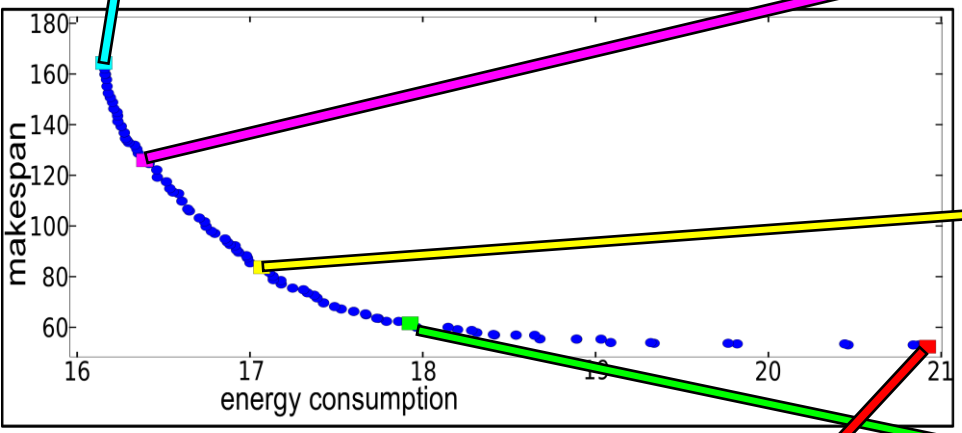


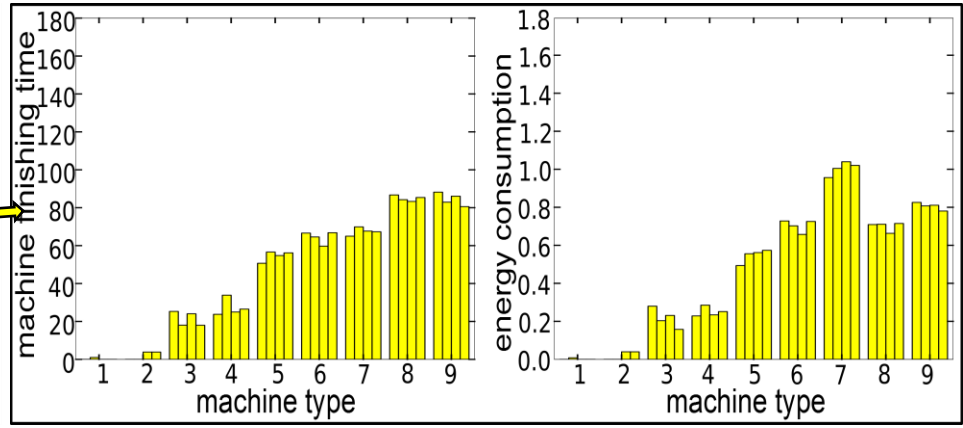


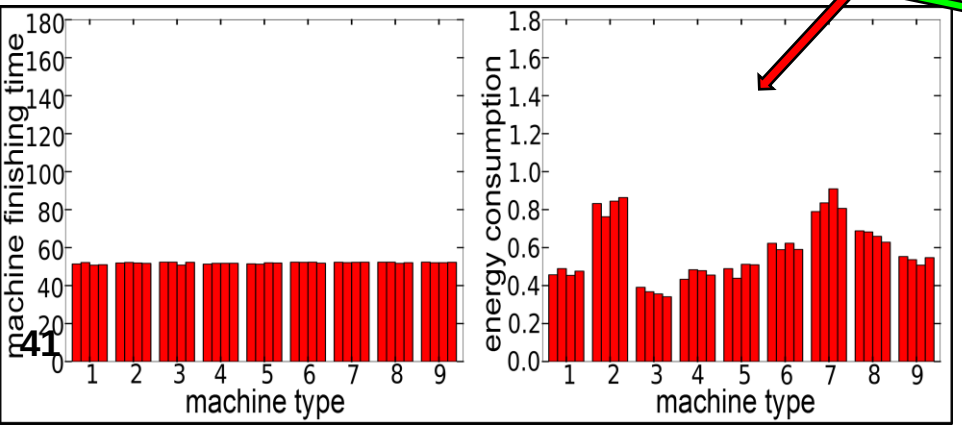
9 Machine Type Environment Analysis

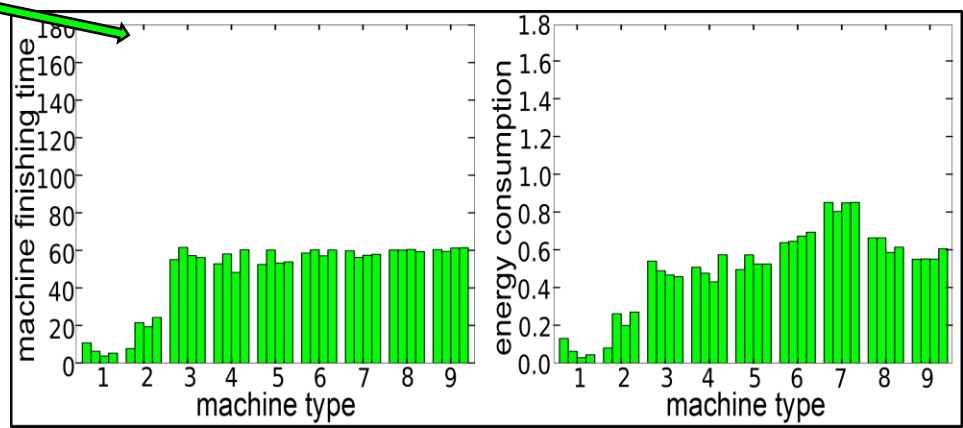












Current Contributions

- modeled a bi-objective resource allocation problem
  - makespan and energy consumed
  - allows for a trade-off analysis
- analyzed on a machine-by-machine basis how different resource allocations affect the behavior on the individual machines
- provided an approach to identify
  - energy-efficient machines
  - energy-inefficient machines
- an analysis approach to perform “what-if” experiments
- showed the versatility of this technique by examining various heterogeneous environments

Possible Future Work

- design new chromosome structure
  - machine types and tasks types
- design new crowding distance measures
  - improved NSGA-II crowding distance (iterative)
- look at new ways to measure distance between chromosomes
  - objective space
  - solution space

Outline

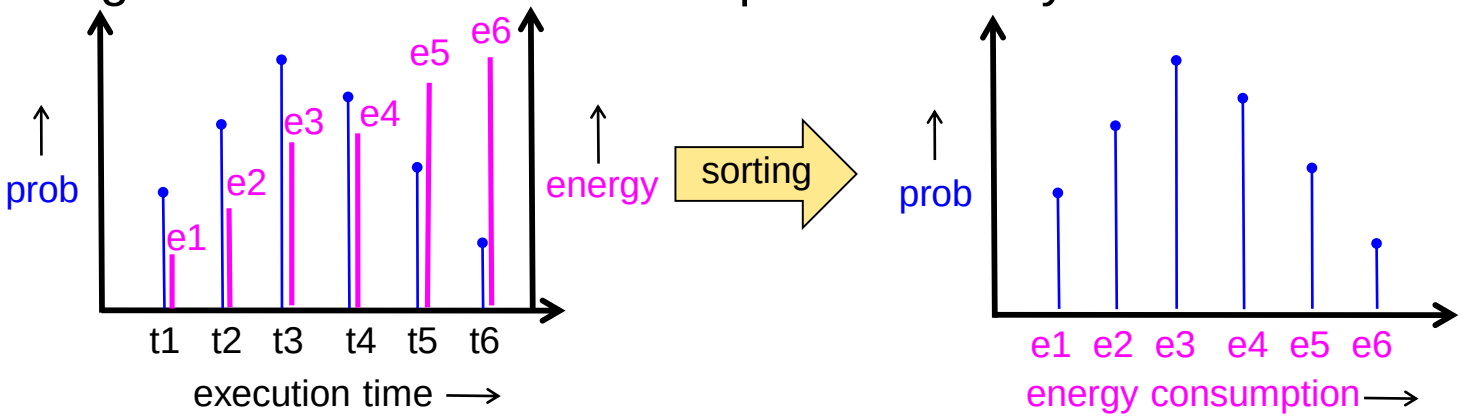
- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

Problem Definition

- given:
  - probability distributions for the ETC task types on machines types in different P-states
    - energy value for each pulse in the ETC distribution
  - utility functions for every task that arrives during the day
  - energy consumption budget for the day (may be scaled from an annual energy constraint)
- maximum attainable utility for a given task: highest utility the task can earn, with non-zero probability, on any machine in any P-state if it starts execution as soon as it arrives (utility at the earliest execution time pulse across all PMFs [machine and P-states] for this task)
- ideal utility: sum of the maximum attainable utility for all tasks
- **constraint:** should meet day's energy budget
- **goal:** maximize the percentage of ideal utility earned while meeting the constraint

Computation for Energy Filtering

- filtering removes all task-machine-P-state choices that consume more energy than a fair-share budget
- for each task in the batch on each machine and P-state, if the probability of energy consumption being greater than the fair-share budget is ≥ *f* % (say *f* = 75), then filter this choice
  - initially we will empirically determine value for *f*
  - long term: we can make it adaptive to the system



- these distributions can be computed before any resource allocations are performed

Handling Tasks that Break the Day's Energy Budget

- if a task breaks the energy budget for a day and therefore has to use energy from the next day, then we want to appropriately allocate the utility earned by the task based on the energy consumed from each day:
- $utility\ applied\ to\ day_{i+1} = \frac{utility\ earned\ by\ task}{energy\ used\ by\ task} * energy\ used\ in\ day_{i+1}$
- $utility\ applied\ to\ day_{i} = utility\ earned\ by\ task - utility\ applied\ to\ day_{i+1}$

Heuristics To Examine

- Max-Max Expected
  - Util per Expected Exec Time,
  - Util per Expected Energy,
  - Utility
- Weighted Expected
  - Util per Expected Exec Time,
  - Util per Expected Energy,
  - Utility
- Max Task Robustness – Max Associated Utility

Environment Models

- queue based environment
  - contains virtual queue
  - contains one pending slot per machine
  - after a task finishes execution, the task in the pending slot starts execution
- queue based environment without pending slots
  - contains virtual queue
  - after a task finishes execution, the task at the head of the virtual queue starts execution
- poll based environment
  - no virtual queue
  - no pending slot
  - after a task finishes execution, the machine sits idle until a task gets assigned during next mapping

Trade-offs Between Models

- **queued with pending slot**: locks in the next task to eliminate idle time but could delay start time of higher utility tasks
  - gets tasks assigned every mapping event
- **poll**: results in the machine being idle but highest utility tasks can be pushed to machine as soon as next mapping event
  - gets task assigned only if machine is idle at beginning of mapping event
- **queued without pending slot**: combines the other two environments, doesn't lock in pending task but allows tasks to begin executing before next mapping event
  - gets tasks assigned every mapping event

High Level Idle Energy Model

- each machine has an associated base (idle) power value associated with it
- to get the base energy of the system for the day we multiply the base power of each machine by the number of seconds in the day
- we subtract the base energy from the total energy constraint to calculate the remaining energy that is available to execute tasks (i.e. dynamic energy)

Possible Future Work

- design and implement “poll-based” heuristics
- design and implement heuristics that utilize stochastic information
- design and implement additional environment models
  - mapping events when a machine becomes idle
- design and implement utility-aware energy filter
- make weighted heuristics and energy filters adaptive
  - remove the need for parameter sweeps

Outline

- utility maximization in resource management
- deterministic energy-aware utility maximization
- multi-objective optimization
  - bi-objective optimization of utility and energy
  - bi-objective optimization of makespan and energy
- stochastic energy-aware utility maximization
- performance, energy, and interference modelling and analysis

Goals of This Research

- analyze the effects of P-states and memory interference on
  - energy consumption
  - system performance
- examine how different classes of applications (differing levels of memory intensity) react to:
  - changes in P-state
  - being co-allocated with another application
- from this data
  - create more accurate energy and performance models
  - classify applications by their memory intensity
  - design models of performance degradation due to memory interference

Workloads to Use

- want to use applications that are diverse in terms of memory access behavior
- may possibly use applications from:
  - spec benchmarks
  - NAS parallel benchmarks
  - **parsec benchmarks**
  - splash-2 benchmarks
  - HPC challenge benchmarks
  - cloudsuite benchmarks

Measuring Interference

- to measure the effects of interference, want to be able to pin applications to individual cores to systematically measure interference
- will run experiments where the number of applications running (at the same time) are different. (e.g. 1 application running up to N applications running)
  - interference should increase as more applications are running, allowing for the measurement of the effects on power and performance
- different mappings of applications to cores can be specific to individual architectures based on number of cores, cache sizes, shared caches, etc.
- study the interference patterns of running memory intensive applications at the same time as CPU intensive applications

Incorporating P-States

- it will be important to understand what effect P-States have on the experiments performed on the previous slide
- depending on the architecture, cores have to all operate in the same P-State, groups of cores can operate in different P-States, or individual cores can operate in different P-States
  - experiments will have to be tailored to the architecture
- initially perform tests where all cores are in same P-State
- possibly perform tests where individual core P-States are assigned based on the memory intensity of the task each core is executing
  - low memory intensity => fast P-State
  - high memory intensity => slow P-State

Acknowledgments

- Portions of This work was supported by Oak Ridge National Laboratory and their Extreme Scale Systems Center, and by the National Science Foundation.

Colorado State University