Efficient and Scalable Computation of the Energy and Makespan Pareto Front for Heterogeneous Computing Systems

Kyle M. Tarplee[1],
Ryan Friese[1], Anthony A. Maciejewski[1],
H.J. Siegel[1,2]

[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science

Colorado State University
Fort Collins, Colorado, USA

2013-09-08

Problem Statement

- static scheduling
  - single bag-of-tasks
  - task assigned to only one machine
  - machine runs one task at a time
- heterogeneous tasks and machines
- desire to reduce energy consumption (operating cost) and makespan
- to aid decision makers
  - find high-quality schedules for both energy and makespan
  - desire computationally efficient algorithms to compute Pareto fronts

Outline

- linear programming lower bound
- recovering a feasible allocation to form an upper bound
- Pareto front generation techniques
- comparison with NSGA-II
- complexity and scalability

Preliminaries

- simplifying approx: **tasks are divisible among machines**
- T_i − number of tasks of type *i*
- M_j − number of machines of type *j*
- x_{ij} − number of tasks of type *i* assigned to machine type *j*
- ETC_{ij} − estimated time to compute for a task of type *i* running on a machine of type *j*
- finishing time of machine type *j* (lower bound):

$$F_j = \frac{1}{M_j} \sum_i x_{ij} ETC_{ij}$$

- APC_{ij} − average power consumption for a task of type *i* running on a machine of type *j*
- $APC_{\emptyset j}$ − idle power consumption for a machine of type *j*

Objective Functions

- makespan (lower bound)

$$MS_{LB} = \max_j F_j$$

- energy consumed by the bag-of-tasks (lower bound)

$$E_{LB} = \text{execution energy} + \text{idle energy}$$

$$= \sum_i \sum_j x_{ij} APC_{ij} ETC_{ij} + \sum_i \sum_j M_j APC_{\emptyset j} (MS_{LB} - F_j)$$

$$= \sum_i \sum_j x_{ij} ETC_{ij} (APC_{ij} - APC_{\emptyset j}) + \sum_j M_j APC_{\emptyset j} MS_{LB}$$

- note that energy is a function of makespan when we have non-zero idle power consumption

Linear Bi-Objective Optimization Problem

$$\underset{x_{ij},\, MS_{LB}}{\text{minimize}} \begin{pmatrix} E_{LB} \\ MS_{LB} \end{pmatrix}$$

subject to:

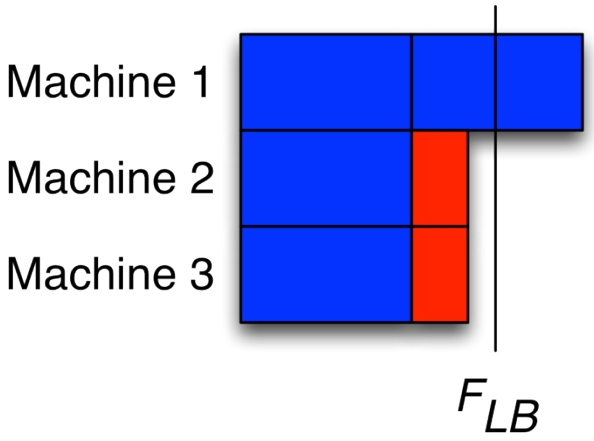
$\forall_i \quad \sum_j x_{ij} = T_i$ task constraint

$\forall_j \quad F_j \le MS_{LB}$ makespan constraint

$\forall_{ij} \quad x_{ij} \ge 0$ assignments must be non-negative

Linear Programming Lower Bound Generation

- solve for $x_{ij} \in \mathbb{R}$ to obtain a lower bound
  - generally infeasible solution to the actual scheduling problem
  - a lower bound for each objective function (tight in practice)
- solve for $x_{ij} \in \mathbb{Z}$ to obtain a tighter lower bound
  - requires branch and bound (or similar)
    - typically this is computationally prohibitive
  - still making the assumption that tasks are divisible among machines



Recovering a Feasible Allocation: Round Near

- find $\dot{x}_{ij} \in \mathbb{Z}$ that is "near" to x_{ij} while maintaining the task assignment constraint
- for task type (each row in x) do
  - let $n = T_i - \sum_j \text{floor}(x_{ij})$
  - let $f_j = \text{frac}(x_{ij})$ be the fractional part of x_{ij}
  - let set K be the indices of the n largest f_j
  - $\dot{x}_{ij} = \text{ceil}(x_{ij})$ if $j \in K$ else $\text{floor}(x_{ij})$

$x_i =$	3	0	6	5	0
$\dot{x}_i =$	3	0	6	5	0

$x_i =$	3	2.3	6.3	5.4	0
$\dot{x}_i =$	3	2	6	**6**	0

$x_i =$	3	0	6.6	5.4	0
$\dot{x}_i =$	3	0	**7**	5	0

$x_i =$	3.9	2.2	6.4	5.3	4.2
$\dot{x}_i =$	**4**	2	**7**	5	4

Recovering a Feasible Allocation: Local Assignment

- given integer number of tasks for each machine type
- assign tasks to actual machines within a homogeneous machine type via a greedy algorithm
- for each machine type (column in $\dot{x}$) do longest processing time algorithm
  - while any tasks are unassigned
    - assign longest task (irrevocably) to machine that has the earliest finish time
    - update machine finish time

Pareto Front Generation Procedure

- **step 1** weighted sum scalarization
  - **step 1.1** find the utopia (ideal) and nadir (non-ideal) points
  - **step 1.2** sweep α between 0 and 1
    - at each step solve the linear programming problem

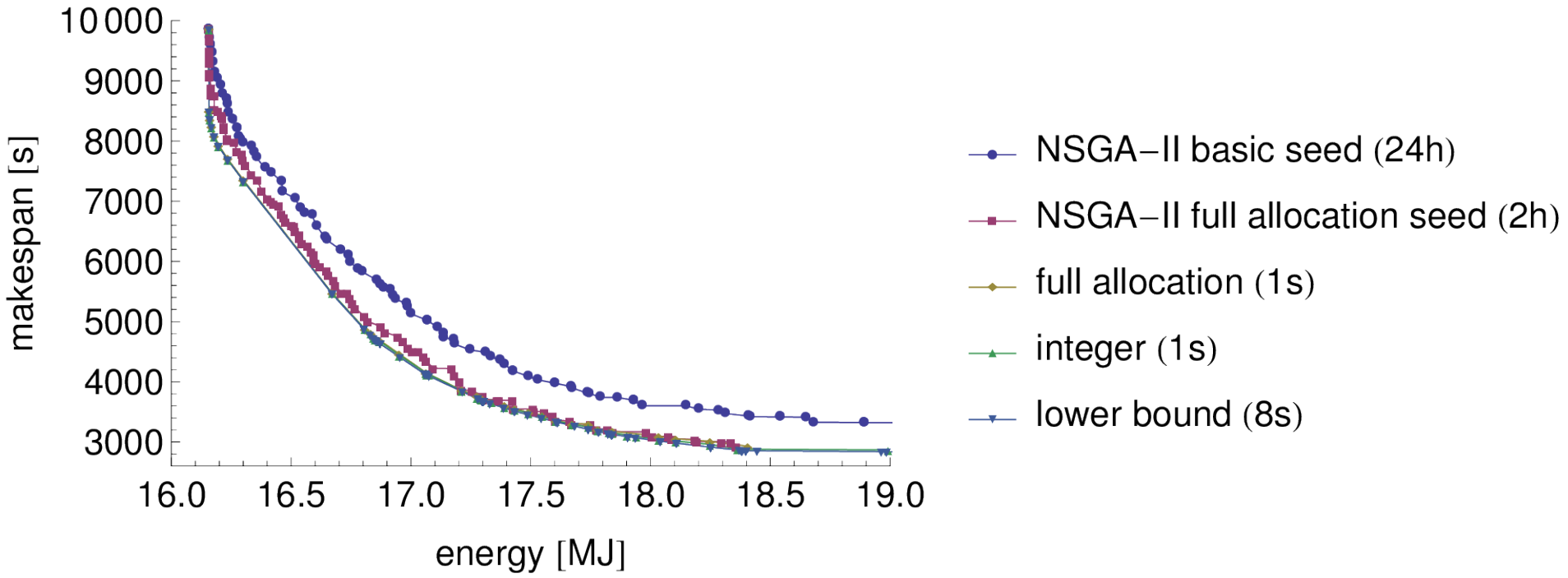
$$\min \frac{\alpha}{\Delta E_{LB}} E_{LB} + \frac{1-\alpha}{\Delta MS_{LB}} MS_{LB}$$

  - **step 1.3** remove duplicates
- linear objective functions and convex constraints
  - convex, lower bounds on Pareto front
- **step 2** round each solution
- **step 3** remove duplicates
- **step 4** locally assign each solution
- **step 5** remove duplicates and dominated solutions
- full allocation is an upper bound on the true Pareto front

Simulation Results

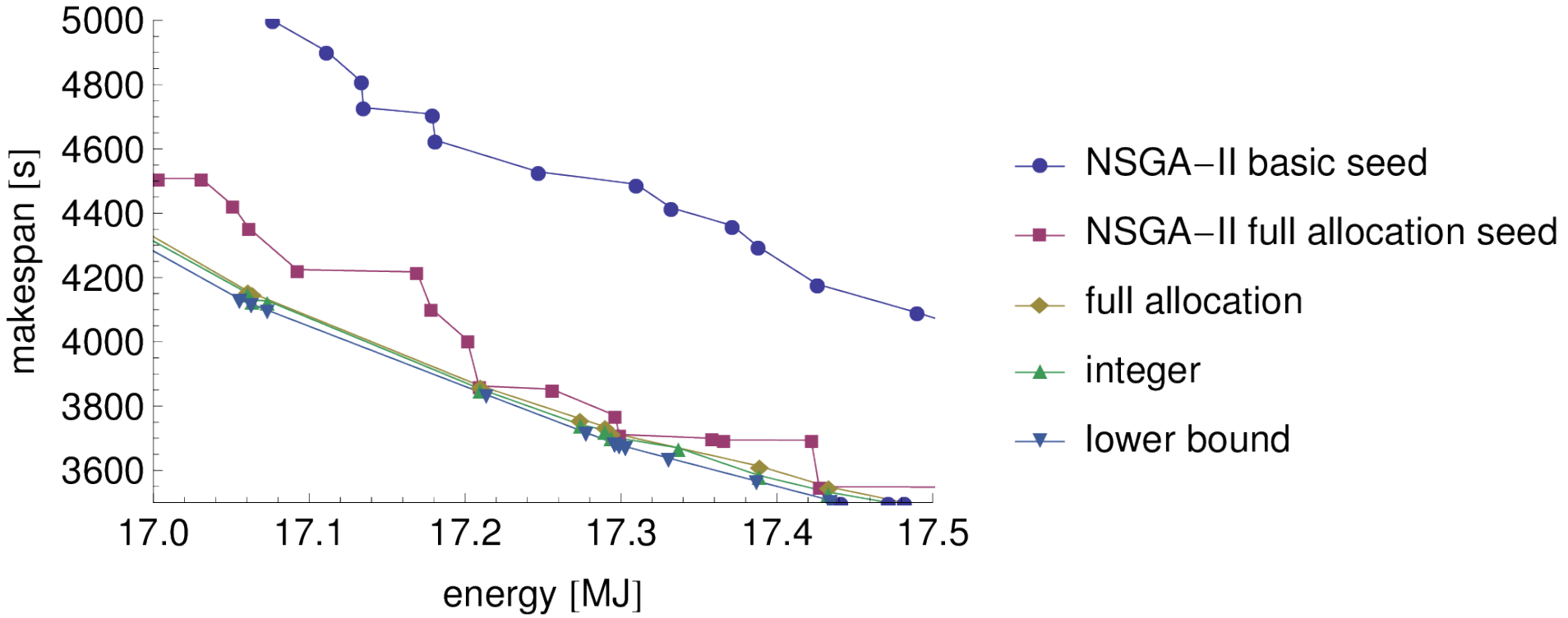
- simulation setup
  - ETC matrix derived from actual systems
  - 9 machine types, 36 machines, 4 machines per type
  - 30 task types, 1100 tasks, 11-75 tasks per type
- Non-dominated Sorting Genetic Algorithm II
  - NSGA-II is another algorithm for finding the Pareto front
  - an adaptation of classical genetic algorithms
  - basic seed
    - min energy, min-min completion time, and random
  - full allocation seed
    - all solutions from upper bound Pareto front

Pareto Fronts



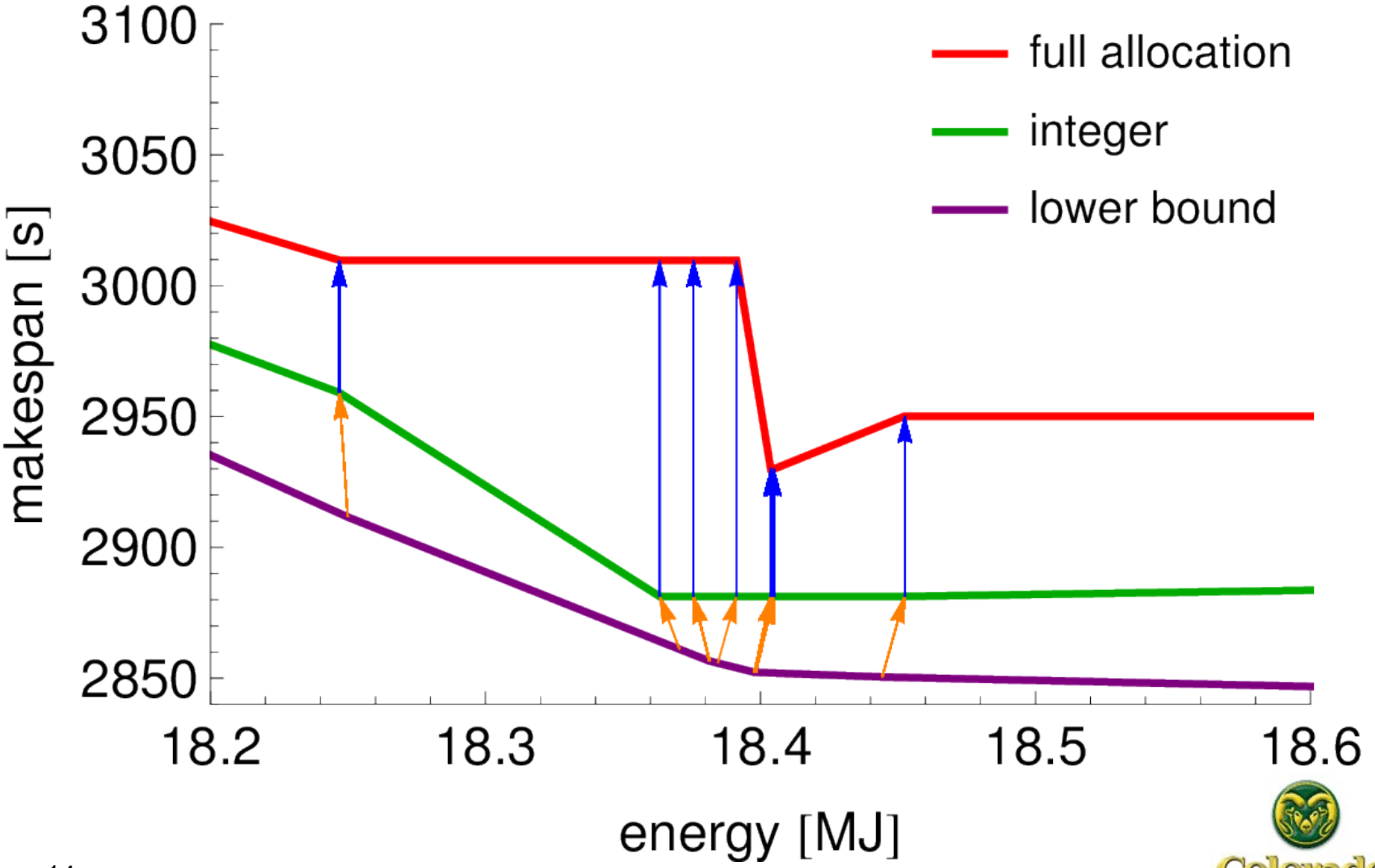
Colorado State University

Pareto Fronts (Zoomed)



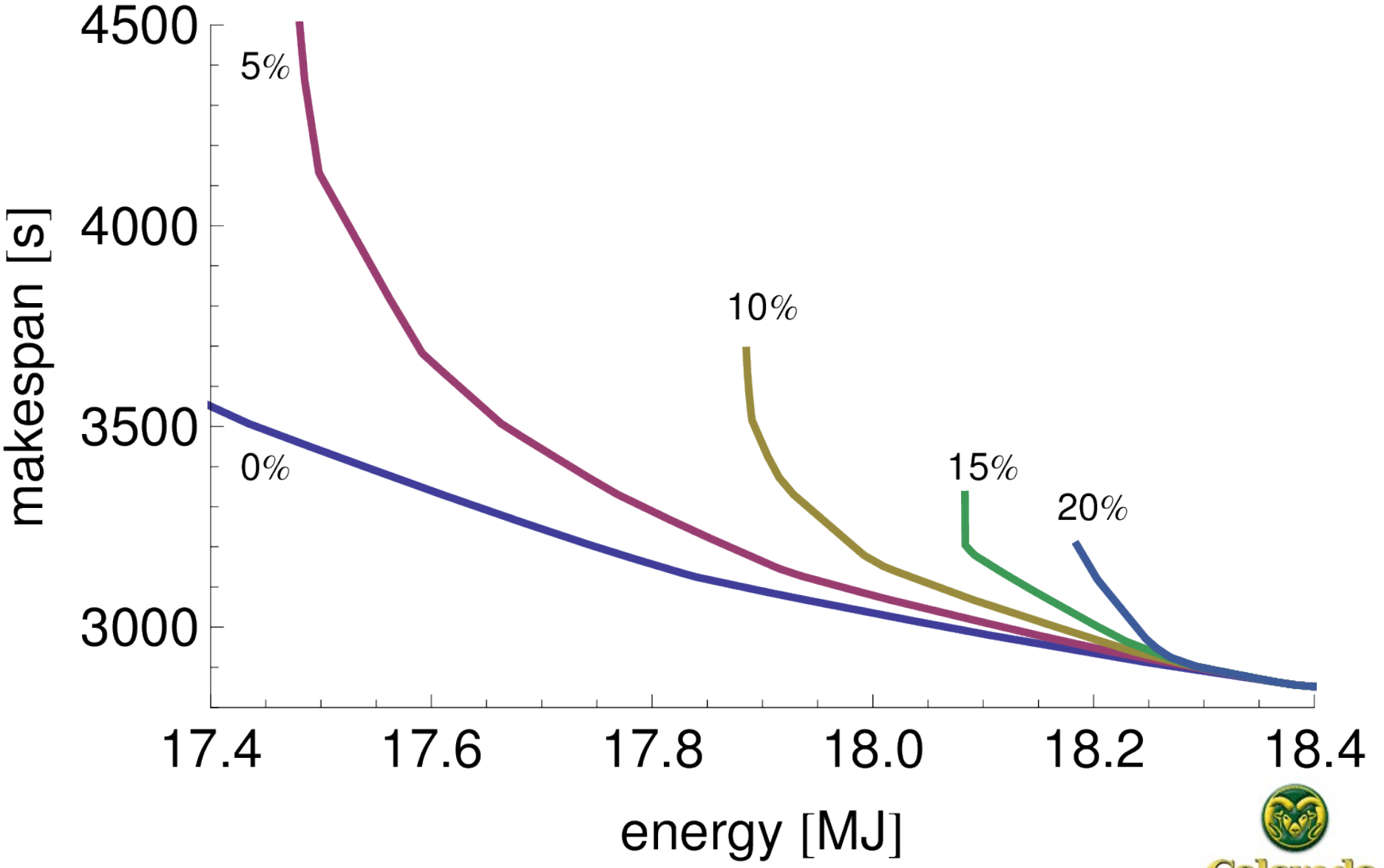
Colorado State University

Lower Bound to Full Allocation, No Idle Power



Colorado State University

Effect of Idle Power

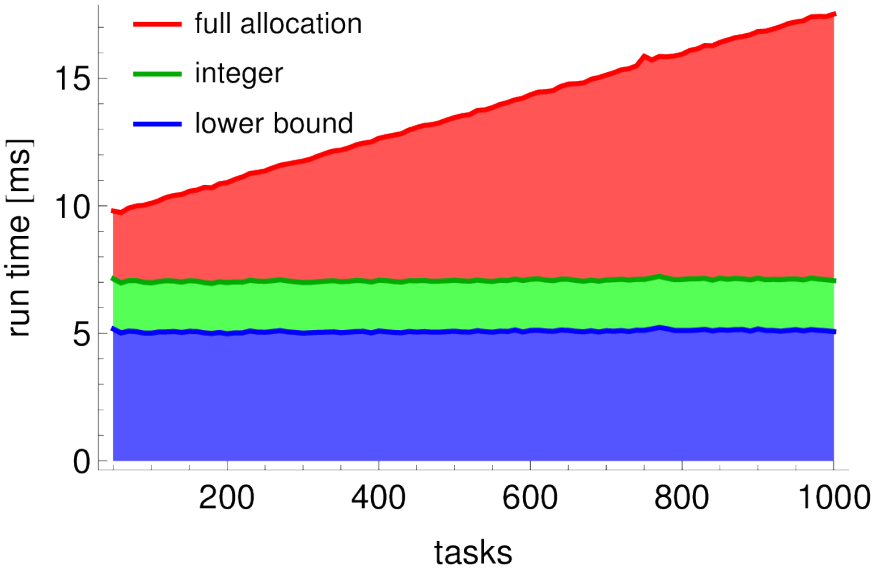


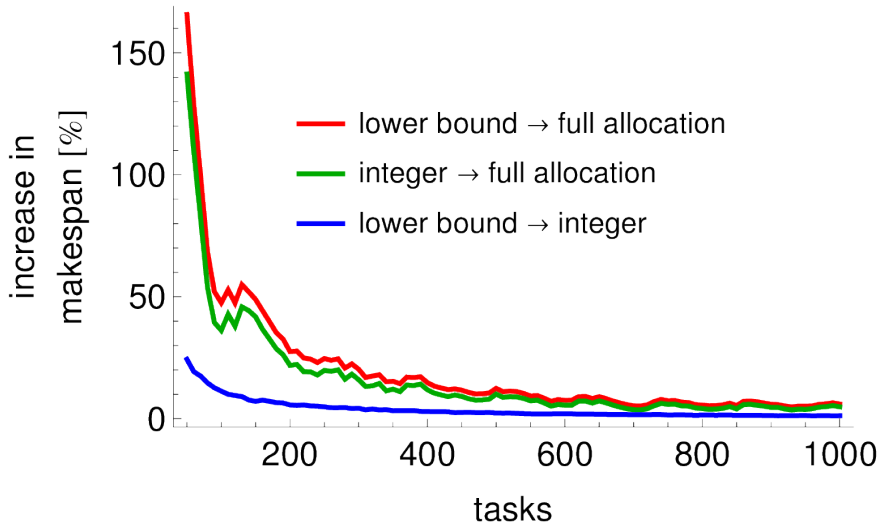
Colorado State University

Complexity

- let the *ETC* matrix be $T \times M$
- linear programming lower bound
  - average complexity $(T+M)^2\ (TM+1)$
- rounding step: $T(M \log M)$
- local assignment step:
  - number of tasks for machine type j is $n_j = \sum_i x_{ij}$
  - worst cast complexity is $M \max_j (n_j \log n_j + n_j \log M_j)$
- complexity of all steps is dominated by either
  - linear programming solver
  - local assignment
- complexity of linear programming solver is **independent** of the number tasks and machines
  - depends number of task types and machine types

Impact of Number of Tasks

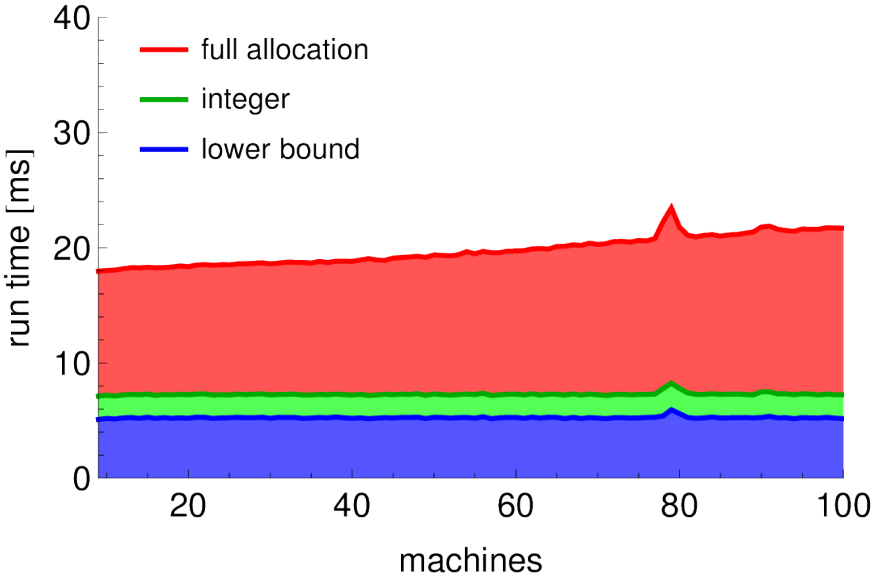


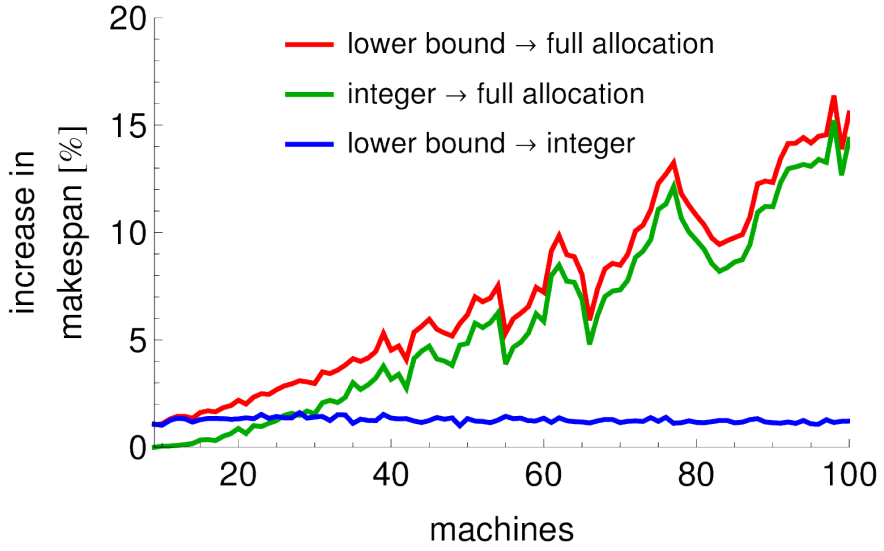


number of task types, machines, and machine types held constant

Colorado State University

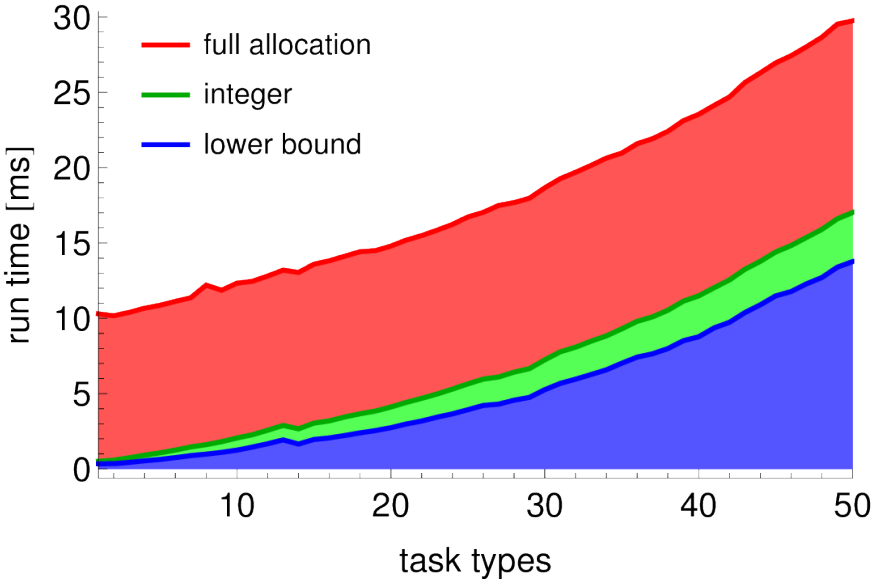
Impact of Number of Machines

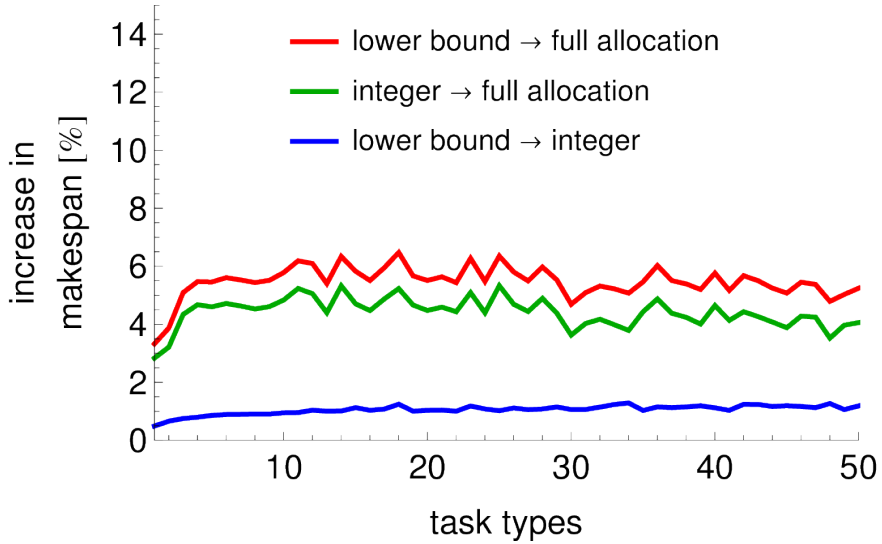




number of tasks, task types, and machine types held constant

Impact of Number of Task Types





number of tasks, machines, and machine types held constant

Colorado State University

Contributions

- algorithms to compute
  - tight lower bounds on the energy and makespan
  - quickly recovers near optimal feasible solutions
  - high quality bi-objective Pareto fronts
- evaluation of the scaling properties of the proposed algorithms

Conclusions

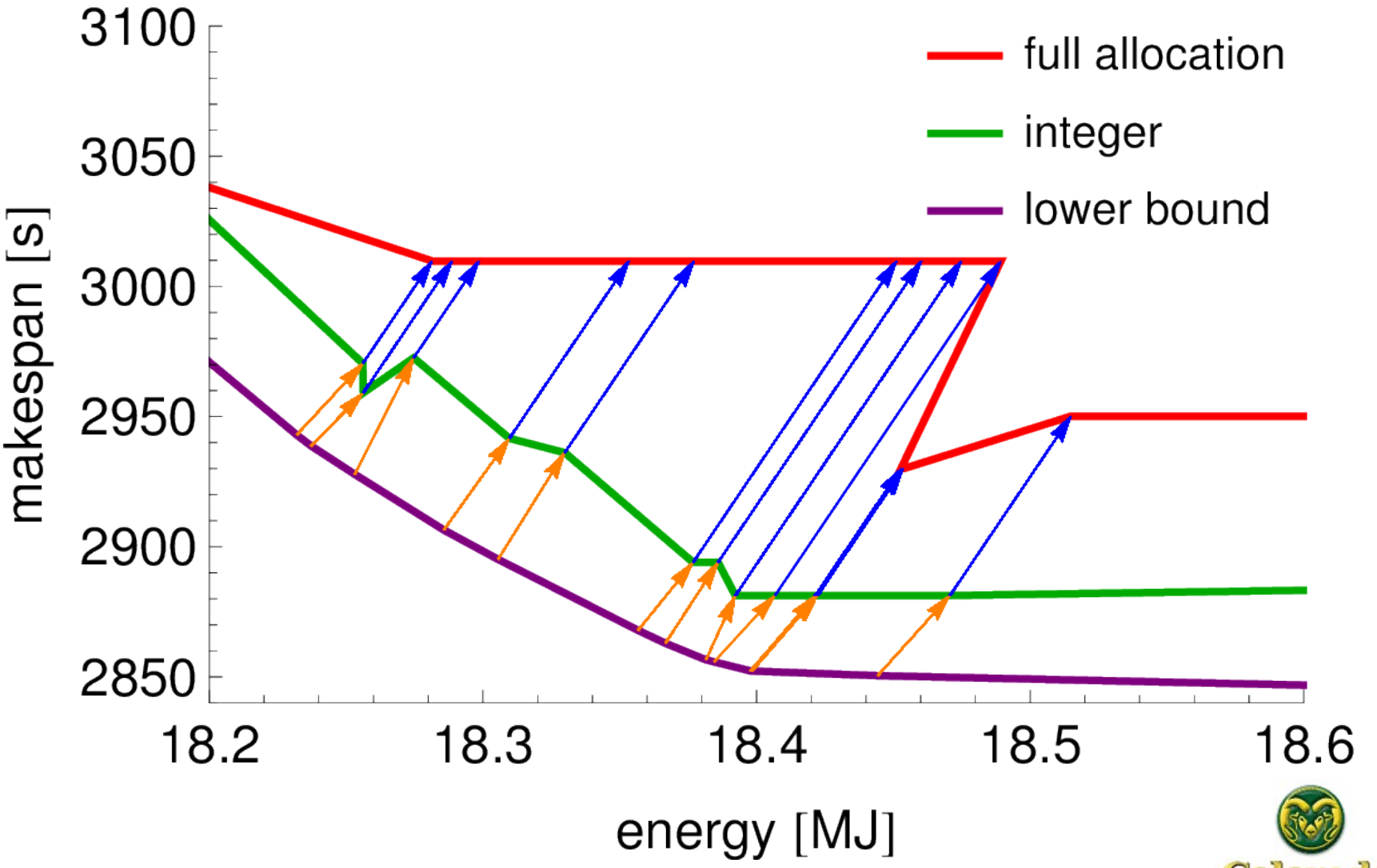
- this linear programming approach to Pareto front generation is efficient, accurate, and practical

- bounds are tight when
  - a small percentage of tasks are divided
  - a large number of tasks assigned to each machine type and individual machine
- asymptotic solution quality and runtime are very reasonable

QUESTIONS?

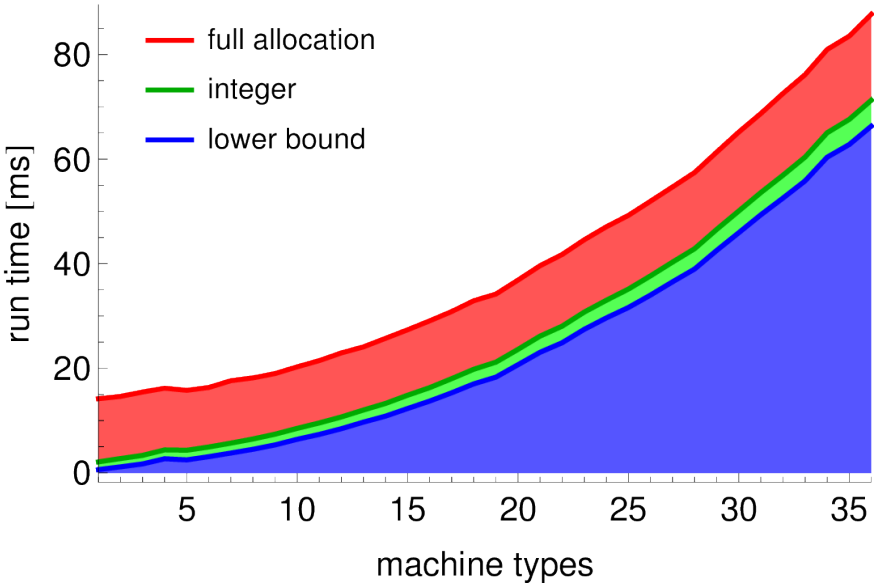
BACKUP SLIDES

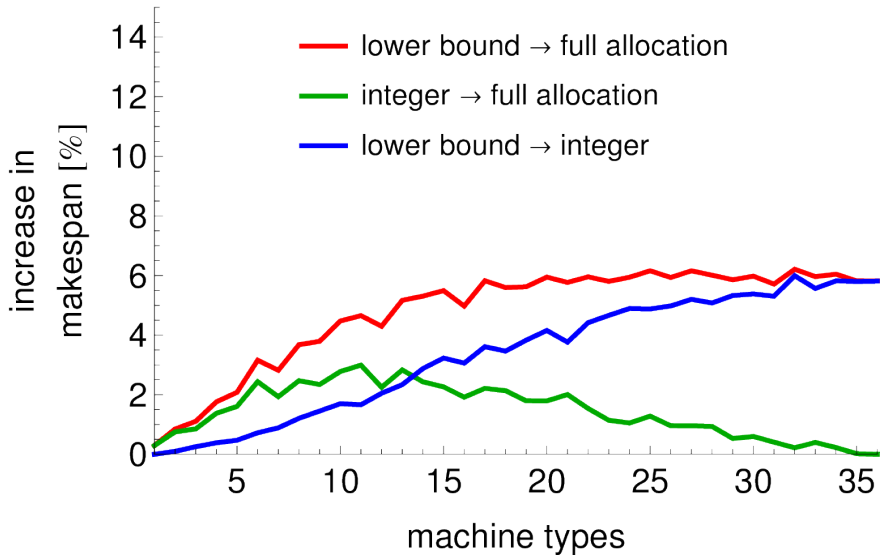
Lower Bound to Full Allocation, 10% Idle Power



Colorado State University

Impact of Number of Machine Types





number of tasks, task types, and machines held constant

Colorado State University