Automatic Extraction of Multi-Objective Aware Parallelism for Heterogeneous MPSoCs

Daniel Cordes, Michael Engel,
Olaf Neugebauer, Peter Marwedel

TU Dortmund University
Department of Computer Science 12
Dortmund
Germany

Introduction

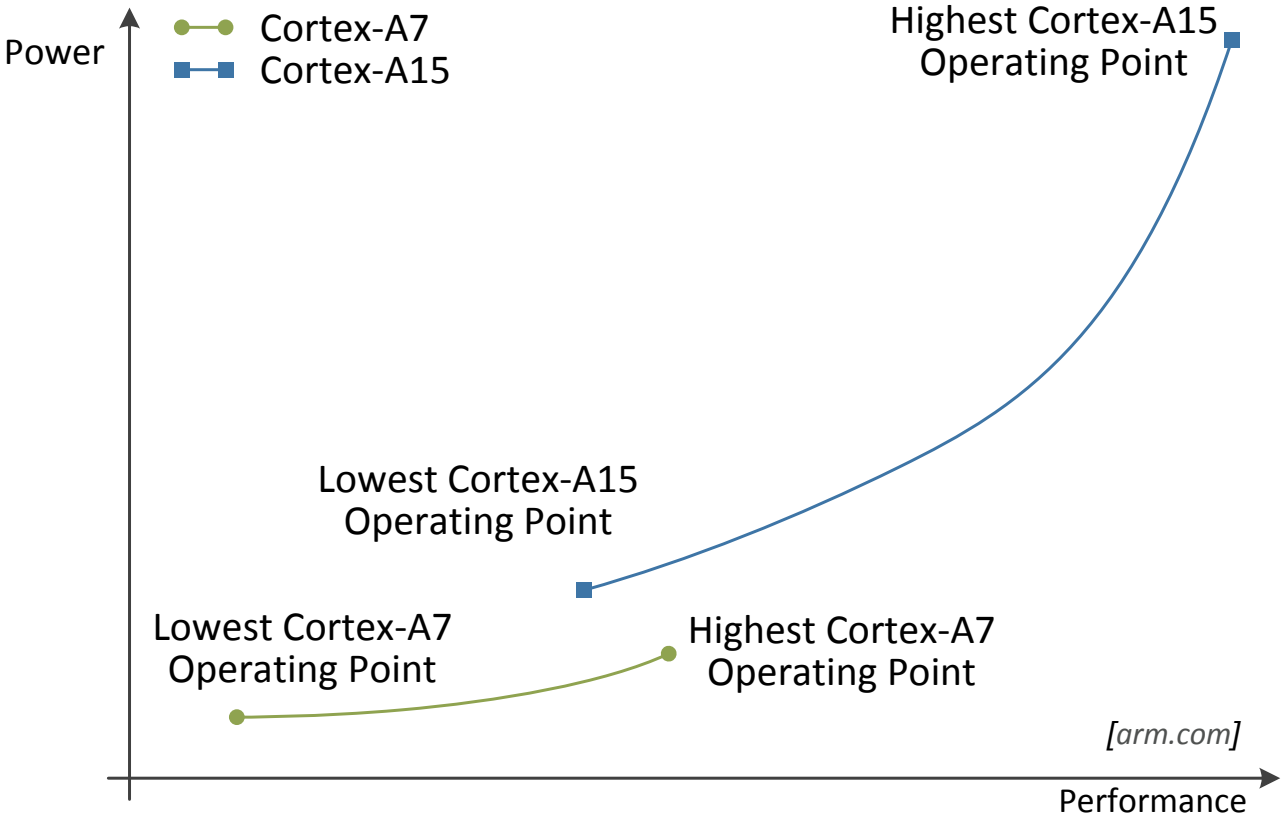
- Complexity of embedded software increasing
  - Multiple cores available in MPSoCs
  - Manual parallelization error-prone & time-consuming
- Heterogeneous MPSoCs can be more efficient than homogeneous
  - Cores behave differently on same parts of application
  - Efficient balancing of tasks very difficult
- Restrictions of embedded architectures
  - Low computational power, small memories, battery-driven
  - Good trade-off between different objectives must be found

➔ Multi-objective aware approach based on Genetic Algorithms to extract parallelism for heterogeneous MPSoCs presented here

Outline

1. Motivating Example
2. Parallelization Methodology
3. GA-based Parallelization Approach
4. Results
5. Conclusion & Future Work

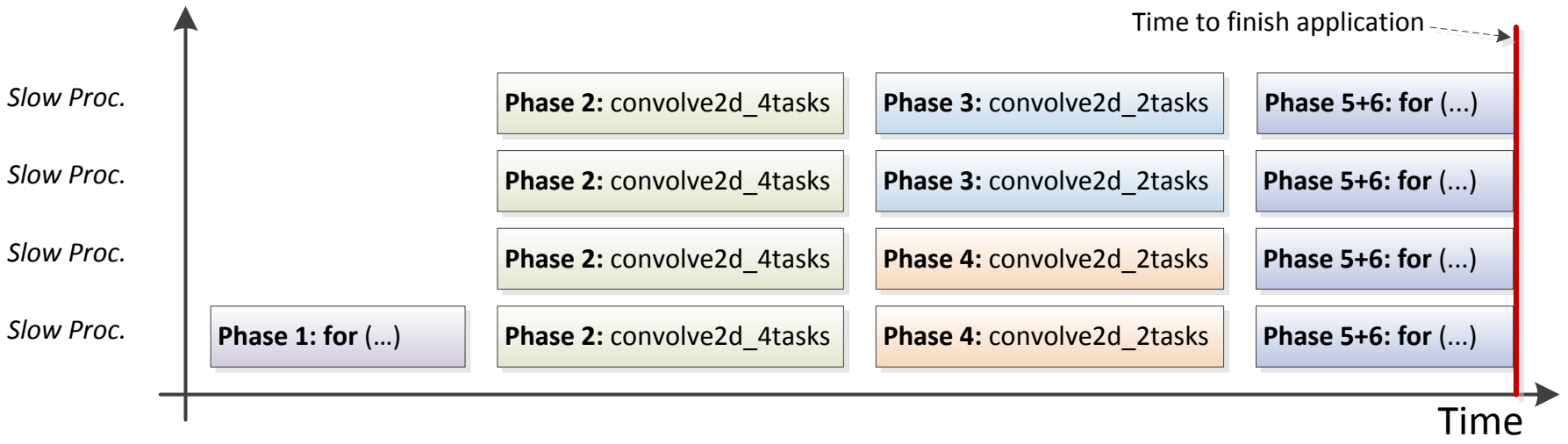
big.LITTLE architecture (ARM)



- Performance vs. power of ARM's big.LITTLE architecture

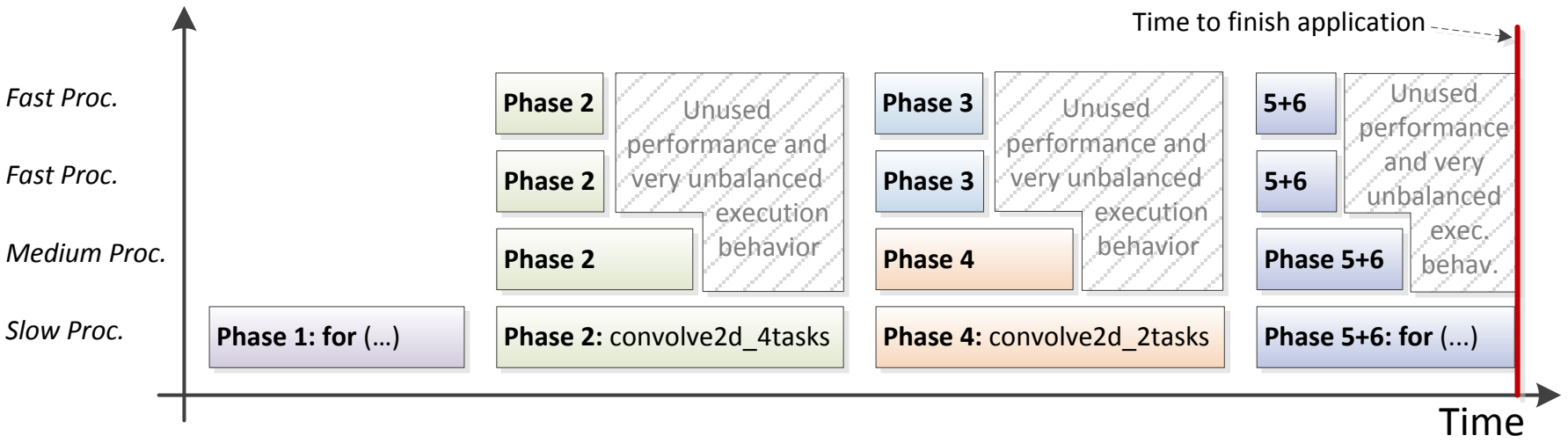
➔ Many trade-offs possible

Example – Homogeneous architecture



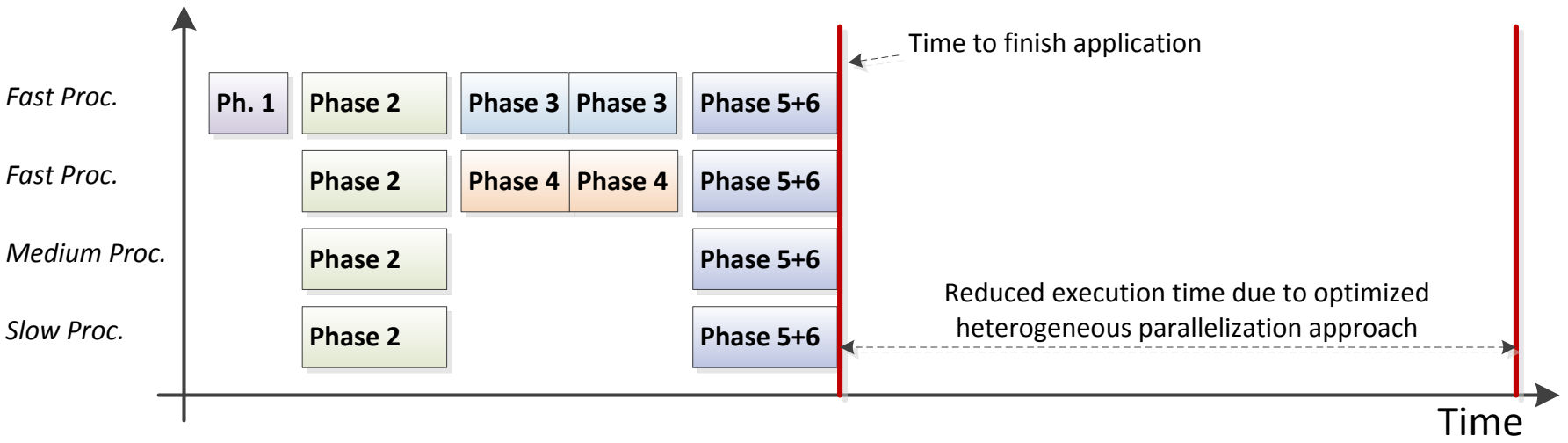
- Timing example of parallelized edge-detect application (UTDSP benchmark suite)
- Execution on homogeneous architecture
- Well-balanced solution

Example – Heterogeneous architecture



- Same solution on heterogeneous architecture
- Faster cores not well utilized
- Unbalanced solution with a lot of wasted performance

Example – Heterogeneous architecture



- Well-balanced solution
- Increased performance of cores utilized
- But: Other solutions might also be better for other objectives!

Question…

How can we extract multi-objective aware parallelism for heterogeneous MPSoCs automatically?

- Input: Sequential C-Code
- Output: Parallelized application
- Used techniques:
  - Annotated Hierarchical Task Graph (AHTG)
  - Genetic Algorithms
  - High-Level Evaluation Models
- Goal:
  - Extract Task and Pipeline Parallelism

[FreeDigitalPhotos.net]

Parallelization Methodology

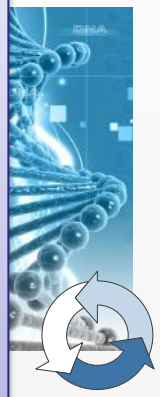
1. Extract ICD-C IR and Augmented Hierarchical Task Graph (AHTG)

ANSI C-Source code

3. Apply parallelization

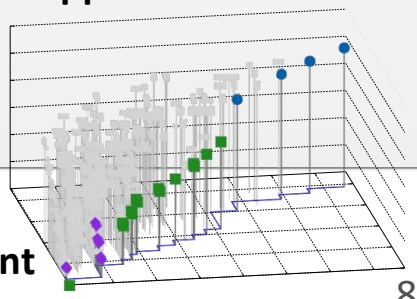
Extraction techniques for homogeneous MPSoCs already exist:

- **DATE'12: Multi-Objective Aware Extraction of Task-Level Parallelism Using Genetic Algorithms.**
- **CODES'12: Automatic extraction of multi-objective aware pipeline parallelism using genetic algorithms.**



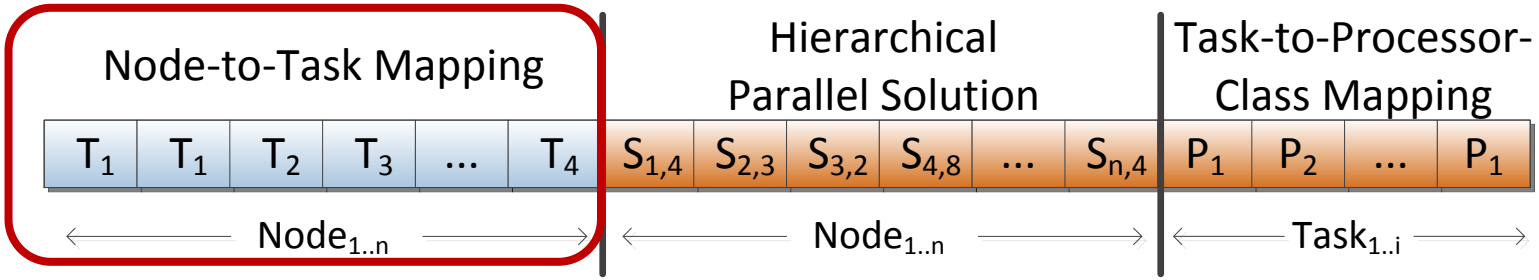
...ront of Pareto-...olution candidates with different approaches

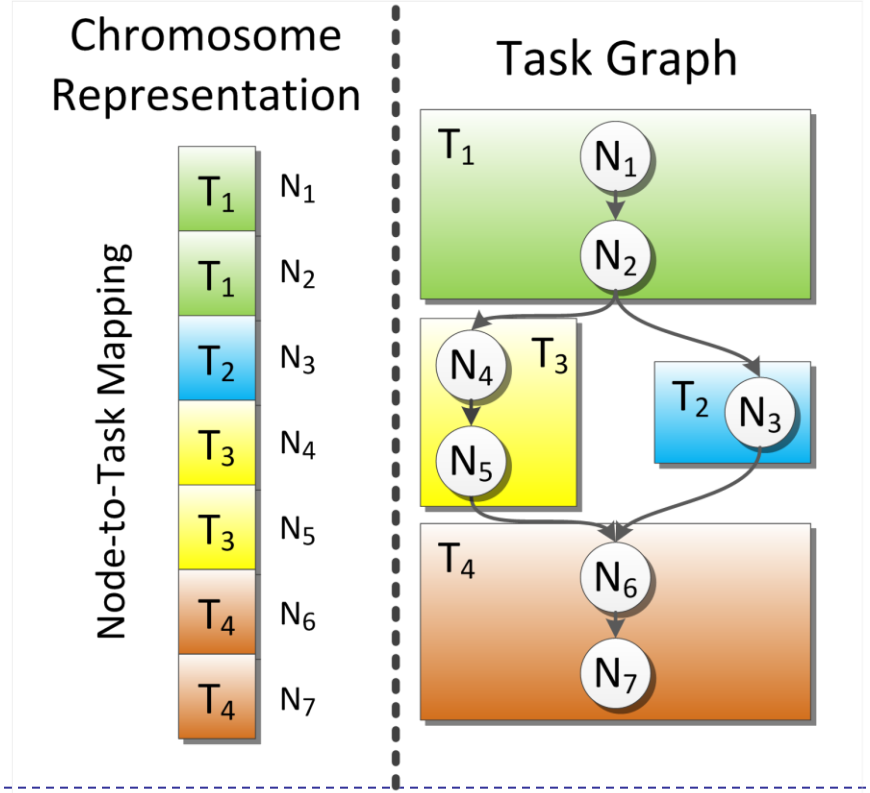
5. Attach solution candidates and continue with other nodes



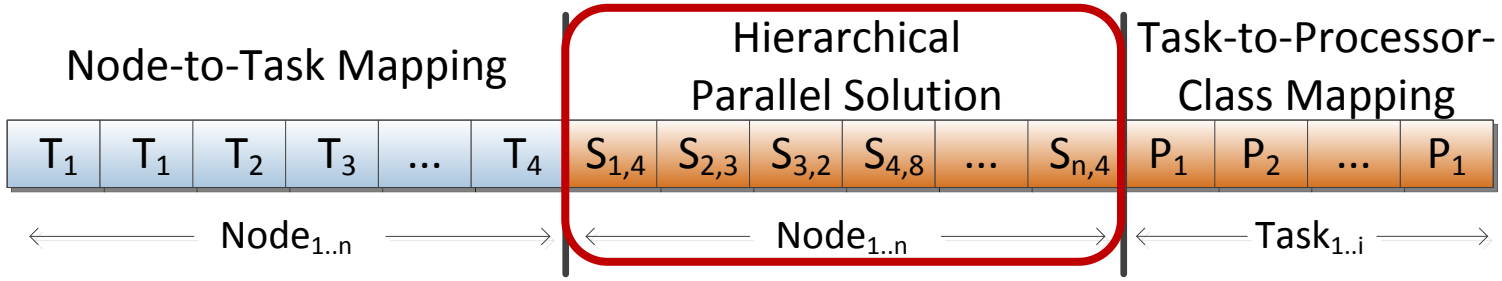
6. Select (best) solution candidate as final result → 7. Annotate Source Code → 8. Implement parallelism

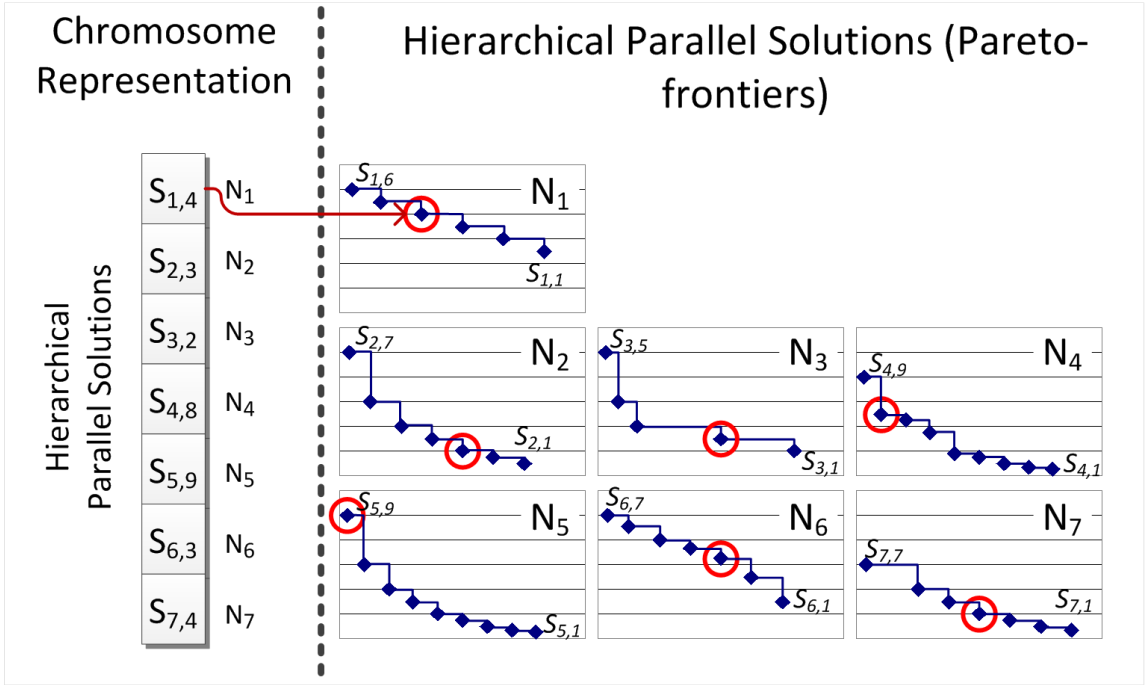
GA-Approach: Task-Level Chromosome

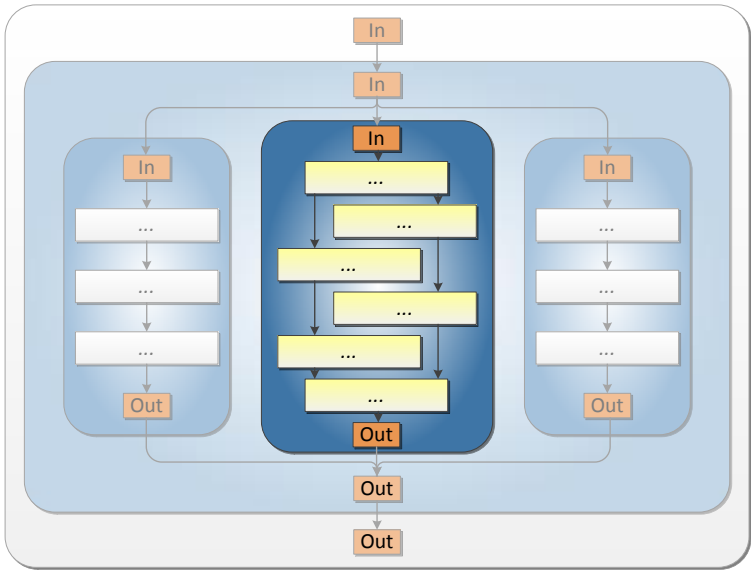




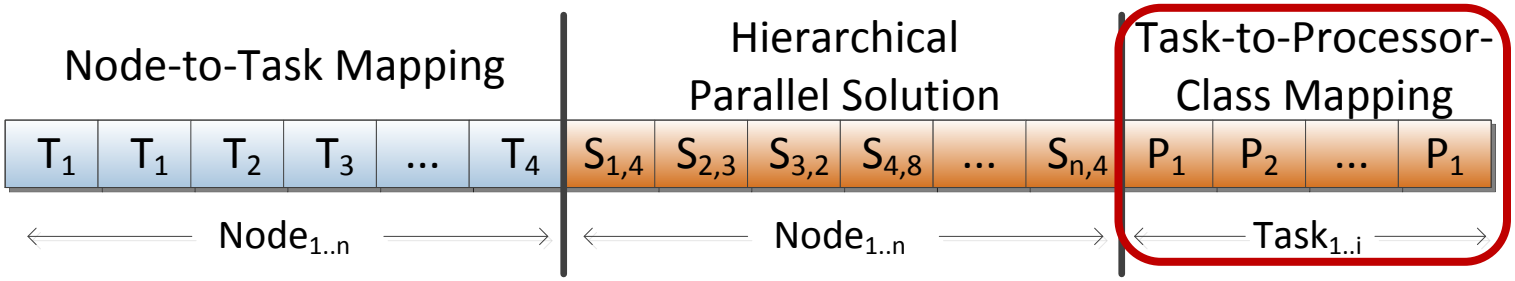
GA-Approach: Task-Level Chromosome

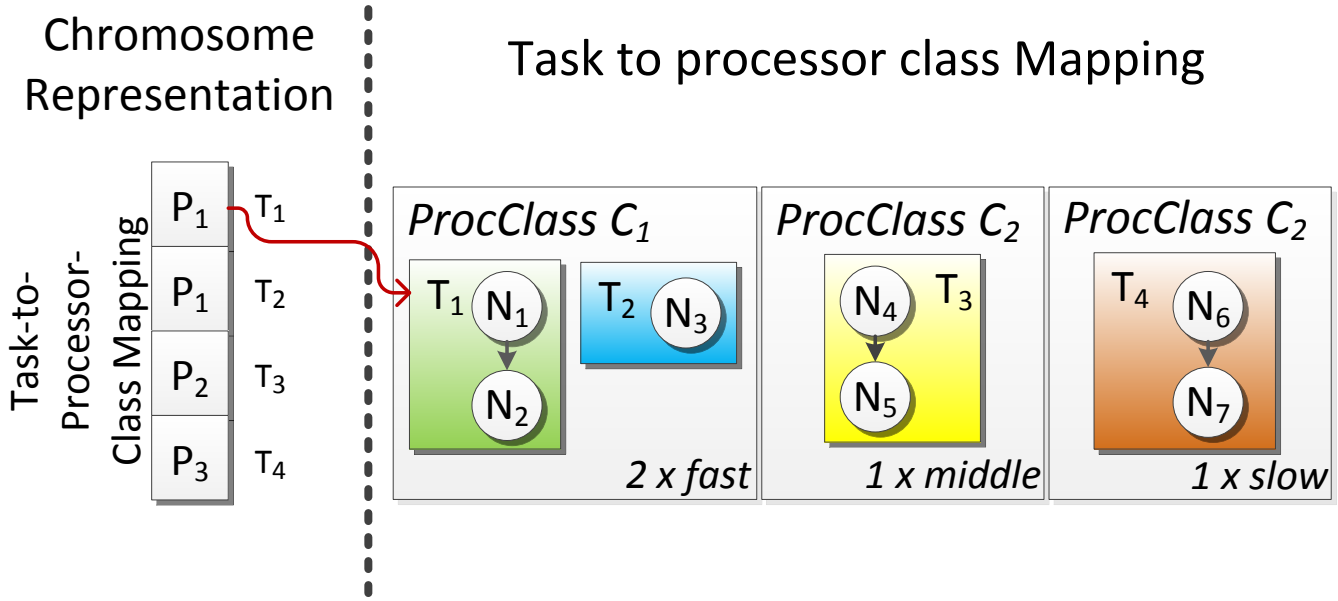




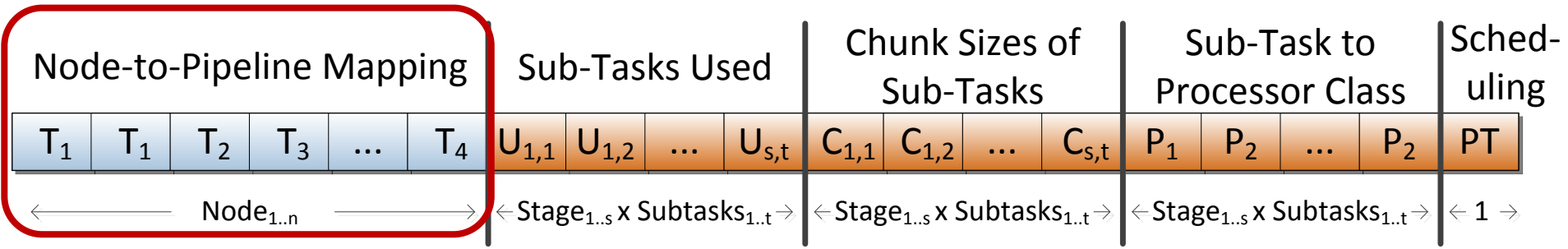


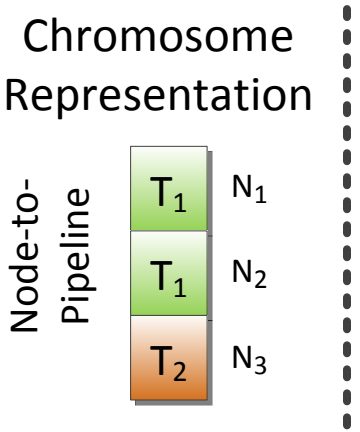
GA-Approach: Task-Level Chromosome

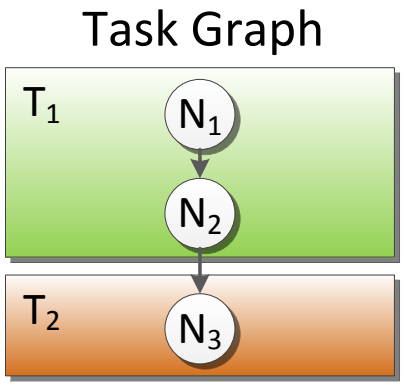




GA-Approach: Pipeline Chromosome

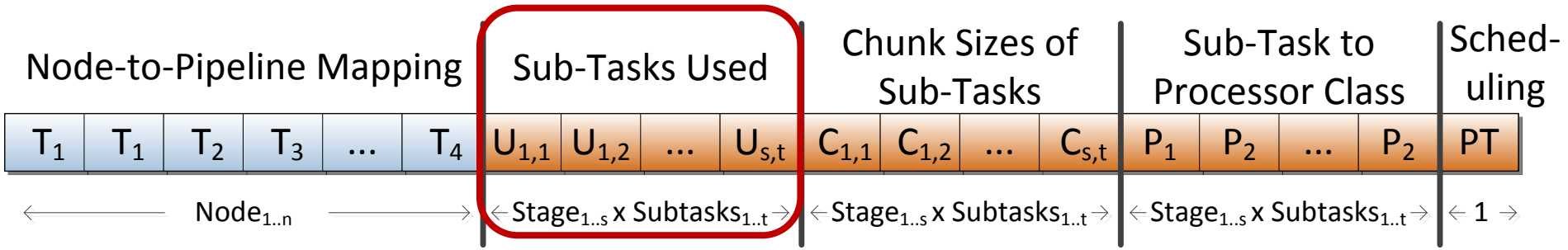




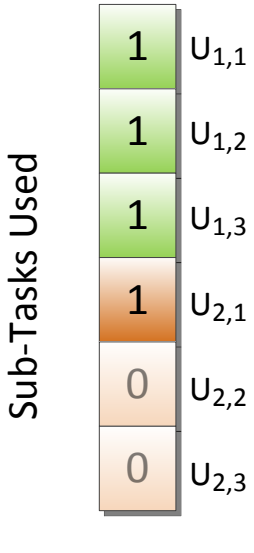


2 Nodes in Stage 1,
1 Node in Stage 2

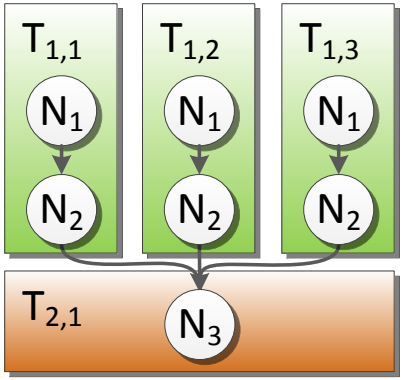
GA-Approach: Pipeline Chromosome



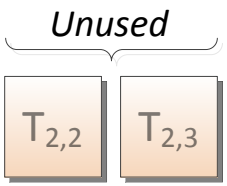
Chromosome Representation



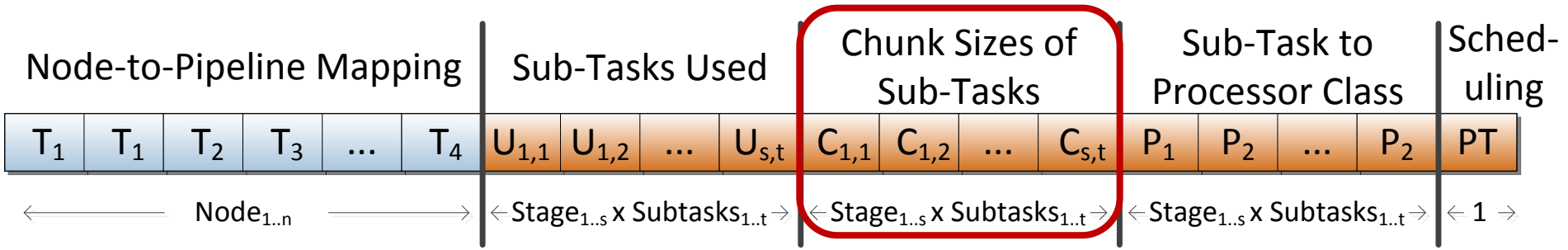
Task Graph

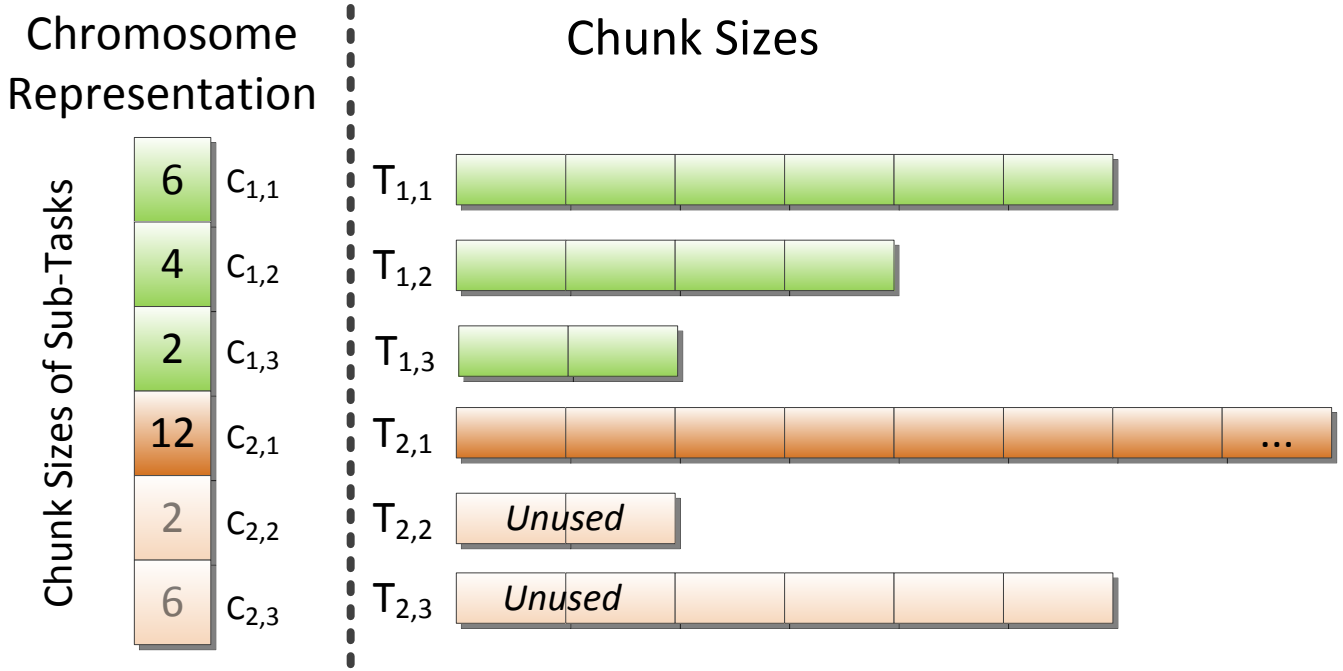


3 Subtasks of Pipeline 1, 1 Subtask of Pipeline 2

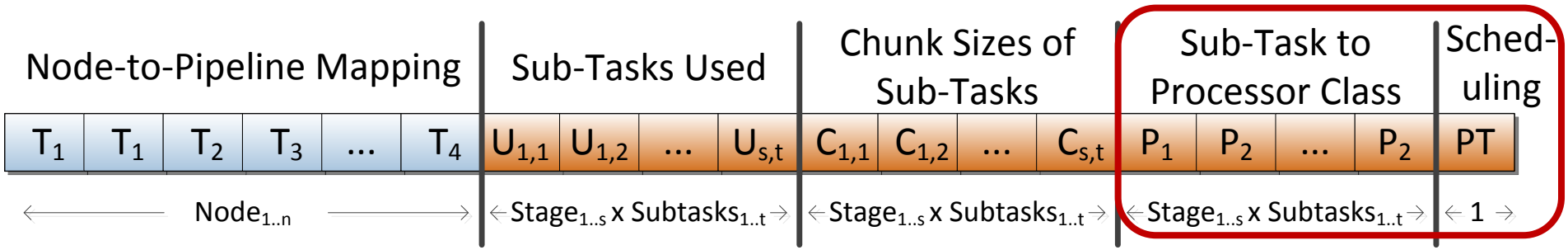


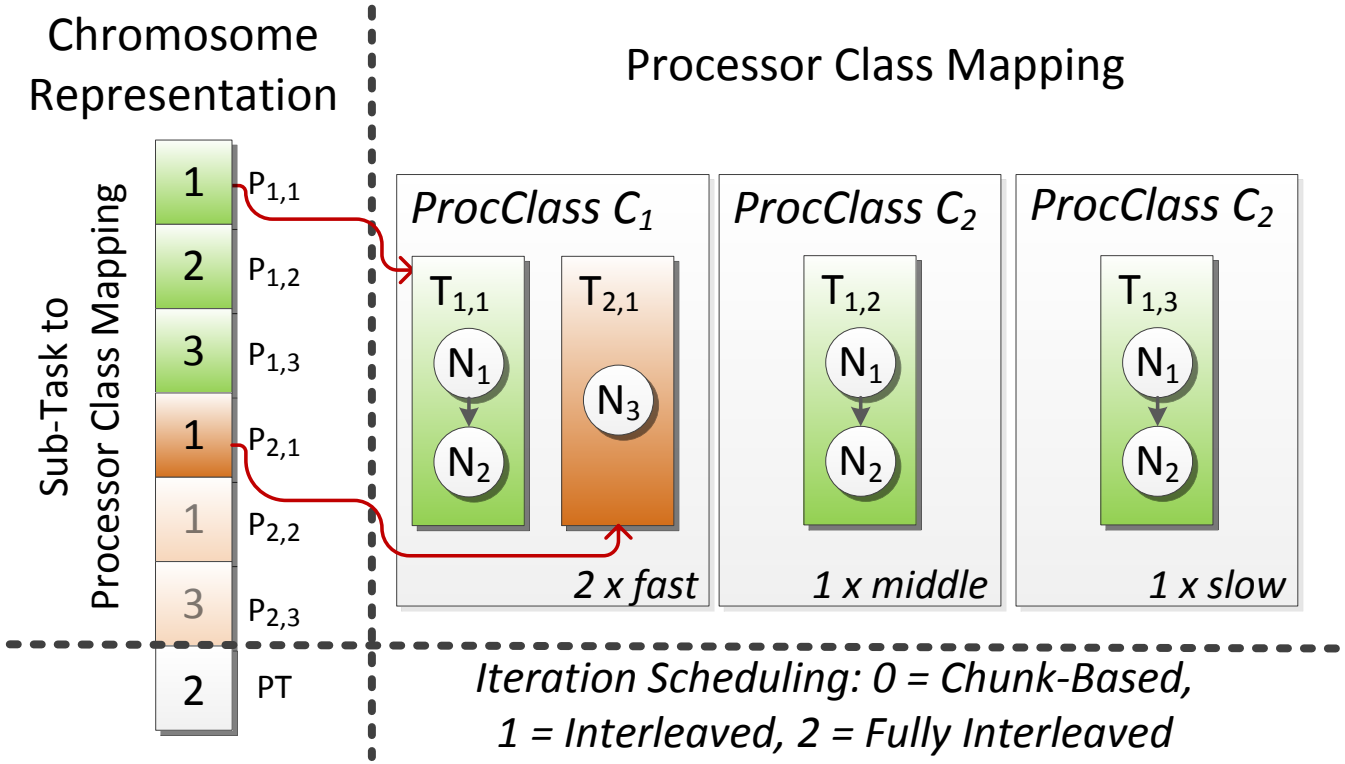
GA-Approach: Pipeline Chromosome



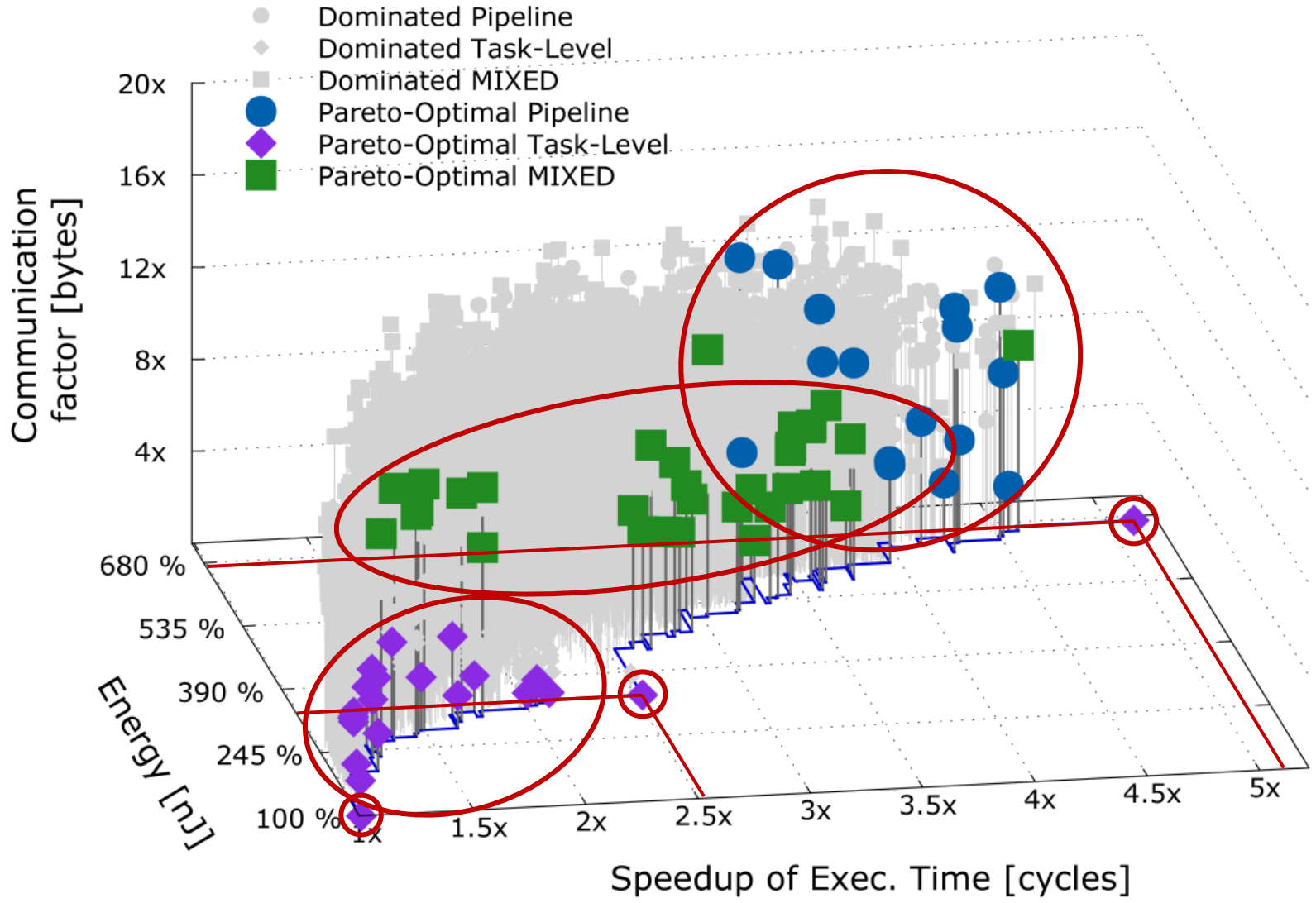


GA-Approach: Pipeline Chromosome





Results – JPEG encoder



GA-Approach: Results - GA-Statistics

Benchmark	Time*	#Nodes	#Populations	#Individuals	#Mutations	#Crossover	#Solutions
adpcm enc.	01:31	36	1,520	153,453	30,729	104,962	63
bound. value	01:17	12	644	83,973	17,900	54,964	34
compress	10:59		8,936	706,266	141,170	464,112	30
edge detect	02:43				0,002	133,096	118
filterbank	03:24				12,229	44,164	47
fir 256	00:30	13	388	29,889	6,629	21,515	44
iir 4				105,224	22,631	79,206	63
jpeg 2000							333
latnrm 32	01:11	17					54
mult 10	01:01	36	1,060	70,855	14,635		90
spectral	**02:25**	**51**	**2,260**	**213,230**	**44,696**	**158,624**	**114**
average	03:01		2,041	179,993	37,293	127,648	90

Most benchmarks processed in 1-2 minutes

213k created and evaluated individuals

Number of solutions range from 43 - 333

Overall 2,5 minutes: 0.6 milli sec. / individual

* AMD Opteron @ 2.4 GHz

Conclusions & Future Work

Conclusion

- New multi-objective aware parallelization approaches for heterogeneous embedded systems presented
- Huge optimization potential
  - High speedups with pipeline technique
  - Less energy consumption with task-level technique
  - Good trace-offs with combination

Future Work

- Integrate DFS/DVS in models
- Integrate other objectives, like, e.g., memory consumption

Thank you for your attention!

Questions?