# Optimal Subset Mapping And Convergence Evaluation of Mapping Algorithms for Distributing Task Graphs on Multiprocessor SoC

Heikki Orsila, Erno Salminen, Marko Hännikäinen, Timo D. Hämäläinen

heikki.orsila, erno.salminen, marko.hannikainen,

timo.d.hamalainen@tut.fi

Institute of Digital and Computer Systems

Tampere University of Technology, Finland

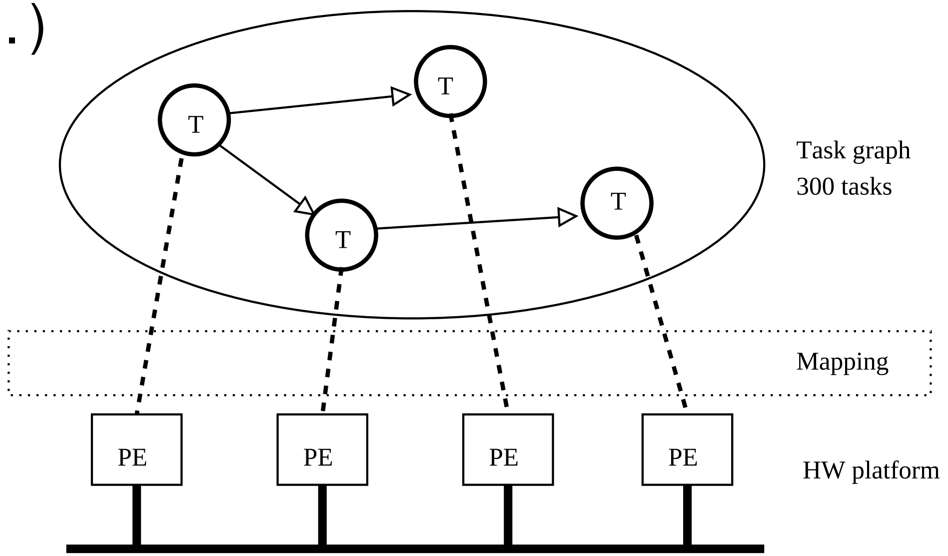
# Presentation Outline

- Introduction
- Experiment
- Algorithms
- Comparison of algorithms
- Conclusions
- References

\+ all the algorithms, graphs and pictures in the end

# Introduction (1/2)

- Automatic distribution of *process networks* onto a multiprocessor system while satisfying some specific criteria
- Assume $N$ tasks in the process network, and $M$ processing elements (PEs) in the multiprocessor system
- Define *mapping* as one possible placement of $N$ tasks to $M$ processing elements (task $i$ goes to PE $x$, task $j$ goes to PE $y$, . . . )



# Introduction (2/2)

- The problem is to minimize a cost function
  - Minimizing the cost function often means maximizing performance or optimizing some other property
  - NP problem $\rightsquigarrow$ true minimum is not generally achieved or known, but *maybe* the result is *good enough*
- Also, try to minimize optimization time
  - $\rightsquigarrow$ trade-off between a good result and short optimization time
  - This is important in exploration of large design space
  - Optimum solution for distribution varies with the architecture

# Contributions

- A new mapping algorithm called *Optimal Subset Mapping* (OSM)
  - OSM sacrifices result goodness to decrease optimization time
- Comparison of mapping algorithms with respect to result goodness, optimization time and converge
- Supporting evidence for our simulated annealing parametrization method presented in [2][3]
- These methods are suitable for both shared and distributed memory systems

# Experiment

- Compare $6$ algorithms
- $10$ random graphs, $N = 300$ nodes
- Simulation run $10$ times independently, results averaged
- $M = 2$, $4$ and $8$ processing elements connected with a shared bus
- Measure speedup with respect to a single processor system

# Algorithms (1/3)

- Use algorithms that have *reasonable* polynomial optimization time upper-bounds with respect to number of tasks $N$ and processing elements $M$
- Upper-bounds for mappings tried for algorithms:
  - Optimal subset mapping (OSM): $O(\frac{N^2 M}{\log N + \log M})$
  - Our simulated annealing variant (SA+AT): $O(NM \log \frac{T_0}{T_f})$
  - Group Migration (GM): $O(N^2 M)$
  - Random mapping: fixed number of iterations (only used as a reference)

# Algorithms (2/3)

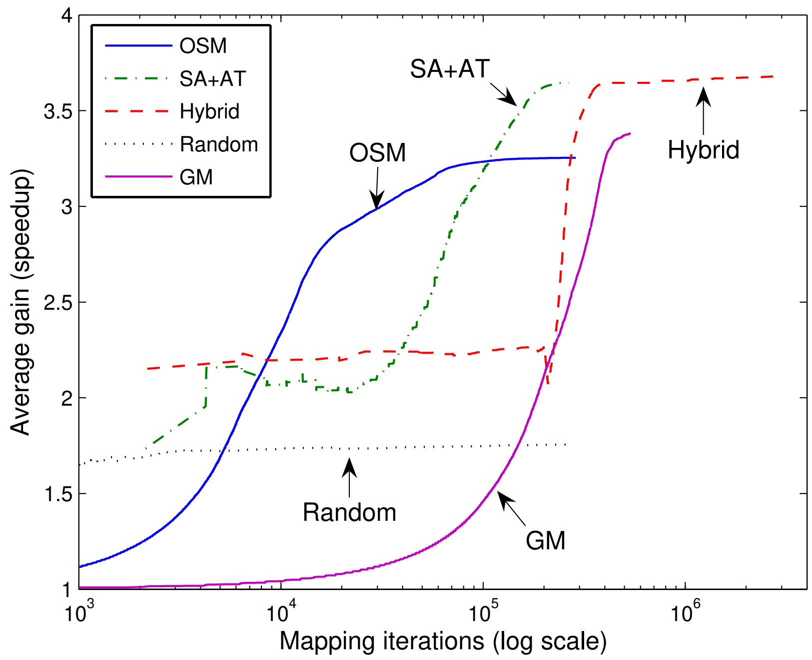
- Group Migration (GM), also known as Kernighan-Lin graph partitioning algorithm
  - deterministic
  - greedy, may get stuck into local minima
- SA+AT is our version of the simulated annealing algorithm [3]
  - Stochastic and non-greedy
  - Automatic temperature (AT) scale is determined from the graph
  - Transition probabilities are normalized for efficient optimization
  - Fully automated parameter selection $\rightsquigarrow$ requires no manual tuning of parameters
- The hybrid algorithm [4] is a combination: result of SA is a starting point for GM

# Algorithms (3/3)

- OSM is the Optimal Subset Mapping algorithm
  - A divide and conquer algorithm. Solves a subset of the problem optimally, but does not guarantee global optimum
  - Picks a subset of tasks and brute-forces an optimal mapping for the subset, and then picks another subset and optimizes that
  - The subset size is increased and decreased continuously when and if there is potential for optimization
  - When increasing the subset size does not improve the result anymore, the algorithm terminates
  - Inspired by the *Sequential Minimal Optimization* algorithm [11] invented for optimizing Support Vector Machine neural networks

# Comparison of Algorithms (1/3)

The following figure shows convergence for $8$ processing elements for each algorithm. The X-axis is the number of mappings tried (logarithmic scale). The Y-axis is the average speedup ($1.0$ means no speedup) over all graphs.




TAMPERE UNIVERSITY OF TECHNOLOGY

# Comparison of Algorithms (2/3)

Following table shows speedups and convergence rate for each algorithm:

| Algorithm | Speedup | Speedup / Iterations | Convergence |
|---|---|---|---|
| Random | $1.76$ | $1.0$ (reference level) | Too long |
| OSM | $3.25$ | $6.11$ | Fast |
| GM | $3.38$ | $1.21$ | Slow |
| SA+AT | $3.65$ | $2.58$ | Fast |
| Hybrid | $3.69$ | $0.20$ | Slow |

# Comparison of Algorithms (3/3)

- Random mapping shows the base-level for optimization
- OSM is most suited for comparing architectures and systems rapidly, but does not yield good speedup
- GM is not suitable for architecture exploration as it is slow and does not yield good speedup
- SA+AT is strong both in convergence speed and speedup $\leadsto$ this is currently our algorithm of choice
- Hybrid algorithm yields the best speedup, but it is slow

Future directions:

- Combine features of each algorithm. For example, start with OSM, and after rapid initial convergence, switch to SA+AT.
- Try genetic algorithms. Problem: hard to select proper parameters

# Discussion

- Almost all papers on task distribution that use Simulated Annealing leave some parameters undocumented
  - Hard to learn about Simulated Annealing even if there are lots of papers that use it
  - We were motivated to document parameters of Simulated Annealing properly [2] [3]
- We use random graphs to avoid application bias in performance
- Static acyclic graphs have very well known scheduling properties, and hence, differences in results are due to mapping algorithms
- Group migration is highly sensitive to initial values, but other algorithms are not

# Conclusions

- This paper demonstrates convergence properties of several algorithms
- This paper demonstrates that automatic parameter selection for simulated annealing can be effective
- SA+AT algorithm converges rapidly, but still yields very good results
- The new OSM algorithm converges very rapidly, but does not yield very good results. It is still suitable for comparing architecture and system alternatives in architecture exploration.

# References (1/3)

1. Y.-K. Kwok and I. Ahmad, *Static scheduling algorithms for allocating directed task graphs to multiprocessors*, ACM Comput. Surv., Vol. 31, No. 4, pp. 406-471, 1999.
2. H. Orsila, T. Kangas, E. Salminen, M. Hännikäinen, T. D. Hämäläinen, *Automated Memory-Aware Application Distribution for Multi-Processor System-On-Chips*, Journal of Systems Architecture, 2007, Elsevier, In print.
3. H. Orsila, T. Kangas, E. Salminen, T. D. Hämäläinen, *Parameterizing Simulated Annealing for Distributing Task Graphs on multiprocessor SoCs*, International Symposium on System-on-Chip (SoC 2006), Tampere, Finland, November 14-16, 2006, pp. 73-76.

# References (2/3)


4. H. Orsila, T. Kangas, T. D. Hämäläinen, *Hybrid Algorithm for Mapping Static Task Graphs on Multiprocessor SoCs*, International Symposium on System-on-Chip (SoC 2005), pp. 146-150, 2005.
5. *Standard task graph set*, [online]: http://www.kasahara.elec.waseda.ac.jp/schedule, 2003.
6. T. D. Braun, H. J. Siegel, N. Beck, *A Comparison of Eleven Static Heuristics for Mapping a Class if Independent Tasks onto Heterogeneous Distributed Systems*, IEEE Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, 2001.
7. G. Kahn, *The semantics of a simple language for parallel programming*, Information Processing, pp. 471-475, 1974.

# References (3/3)


8. T. Kangas, P. Kukkala, H. Orsila, E. Salminen, M. Hännikäinen, T.D. Hämäläinen, J. Riihimäki, K. Kuusilinna, *UML-based Multi-Processor SoC Design Framework*, Transactions on Embedded Computing Systems, ACM, 2006.
9. B.W. Kernighan, S. Lin, *An Efficient Heuristics Procedure for Partitioning Graphs*, The Bell System Technical Journal, Vol. 49, No. 2, pp. 291-307, 1970.
10. S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, *Optimization by simulated annealing*, Science, Vol. 200, No. 4598, pp. 671-680, 1983.
11. J. Platt, *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, Microsoft Research Technical Report MSR-TR-98-14, 1998.

# Optimal Subset Mapping Pseudo-code

```
OPTIMAL_SUBSET_MAPPING(S)
S_best ← S
C_best ← COST(S)
X ← 2
for R ← 1 to ∞
do C_old_best ← C_best
    S ← S_best
    Subset ← PICK_RANDOM_SUBSET(S, X)
    for all possible mappings S_sub in Subset
    do S ← APPLY_MAPPING(S, S_sub)
        C ← COST(S)
        if C < C_best
          then S_best ← S
                C_best ← C
    if modulo(R, R_max) = 0
      then if C_best = C_old_best
              then if X = X_max
                      then break
                    X ← X + 1
              else  X ← X − 1
            X ← MAX(X_min, X)
            X ← MIN(X_max, X)
return S_best
```

# Application and architecture parameters

| | Variable | (note) | Value |
|---|---|---|---|
| Task graphs | # graphs | | 10 |
| | # tasks per graph ($N$) | | 302 |
| | # edges per graph | (1) | 1594, 5231, 8703 |
| | comp time per task [us] | (1) | 3.2, 5.1, 7.0 |
| | comm vol per task [byte] | (1) | 26, 1111, 3679 |
| | comm/comp -ratio [Mbyte/s] | (1) | 8, 218, 526 |
| | max theor. parallelism [no unit] | (1) | 4.3, 7.9, 12.8 |
| HW Platform | # PEs ($M$) | | 2, 4, 8 |
| | PE freq [MHz] | | 50 |
| | Bus Freq [MHz] | (2) | 10, 20, 40 |
| | Bus width [bits] | | 32 |
| | Bus bandwidth [Mb/s] | (2) | 320, 640, 1280 |
| | Bus arb. latency [cycles/send] | | 8 |
| Algorithms | # runs per graph per alg | (3) | 10 |
| | algorithms | | 6 |
| | determ, non-greedy | | 1: OSM |
| | determ, greedy | | 1: GM |
| | stoch., non-greedy | | 4: SA, SA+AT, hybrid, random |
| | stoch, greedy | | - |

Notes:

(1) = min, avg, max

(2) = values for 2,4,8 PEs, respectively

(3) = only 1 run for GM

# Optimization parameters

| Alg. | Variable (note) | Value |
|---|---|---|
| SA, SA+AT, Hybrid | # iter per $T$, ($L = N \cdot (M-1)$) (1) | 602, 1208, 2416 |
| | # temperature levels | 181 |
| | # temperature scaling | $q$=0.95 |
| | range of T (SA and hybrid) (2) | $T_0$= 1.0, $T_f$=0.0001 |
| | range of T (SA+AT) | $T$ range coefficient $k$=2 |
| | *annealing schedule* ($T_0$, $i$) | $T_0 \cdot q^{floor(i/L)}$ |
| | move heuristic | move 1 random task |
| | acceptance function | $(1 + \exp(\Delta C \,/\, (0.5\, C_0\, T))^{-1}$ |
| | end condition | $T = T_f$ AND $L$ rejected moves |
| Rand | # max interations | 262 144 |
| GM | no params needed | - |
| OSM | coefficient $c$ | 1.0 |
| | exponent $c_N$ | 1.0 |
| | exponent $c_M$ | 1.0 |
| | subset size $X$ [#tasks] (1) | 9, 5, 3 |
| | # iterations per subset (1) | 512, 1024, 512 |

Notes:

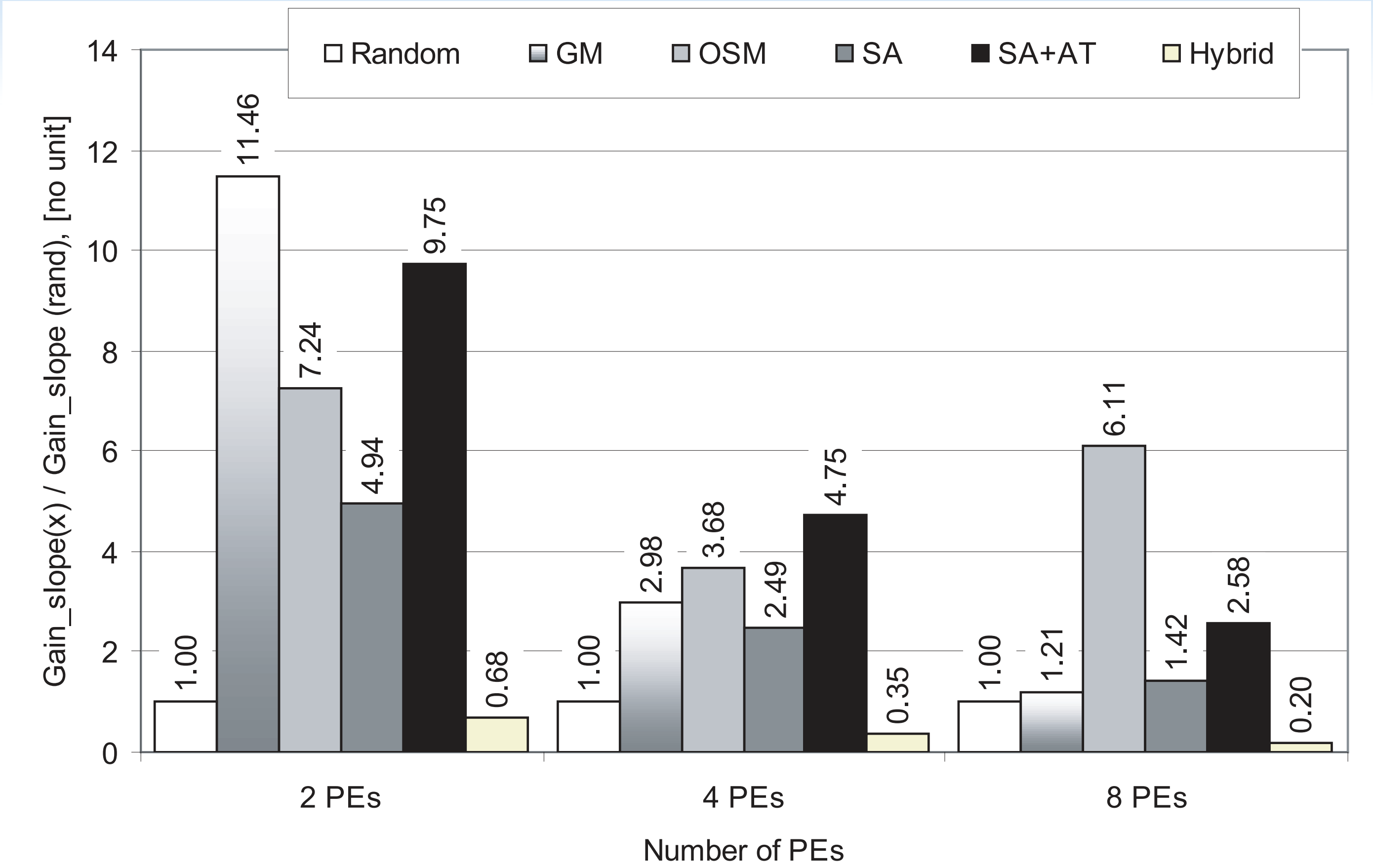
(1) = values for 2,4,8 PEs, respectively

(2) = $T_0$ and $T_f$ computed automatically in SA+AT

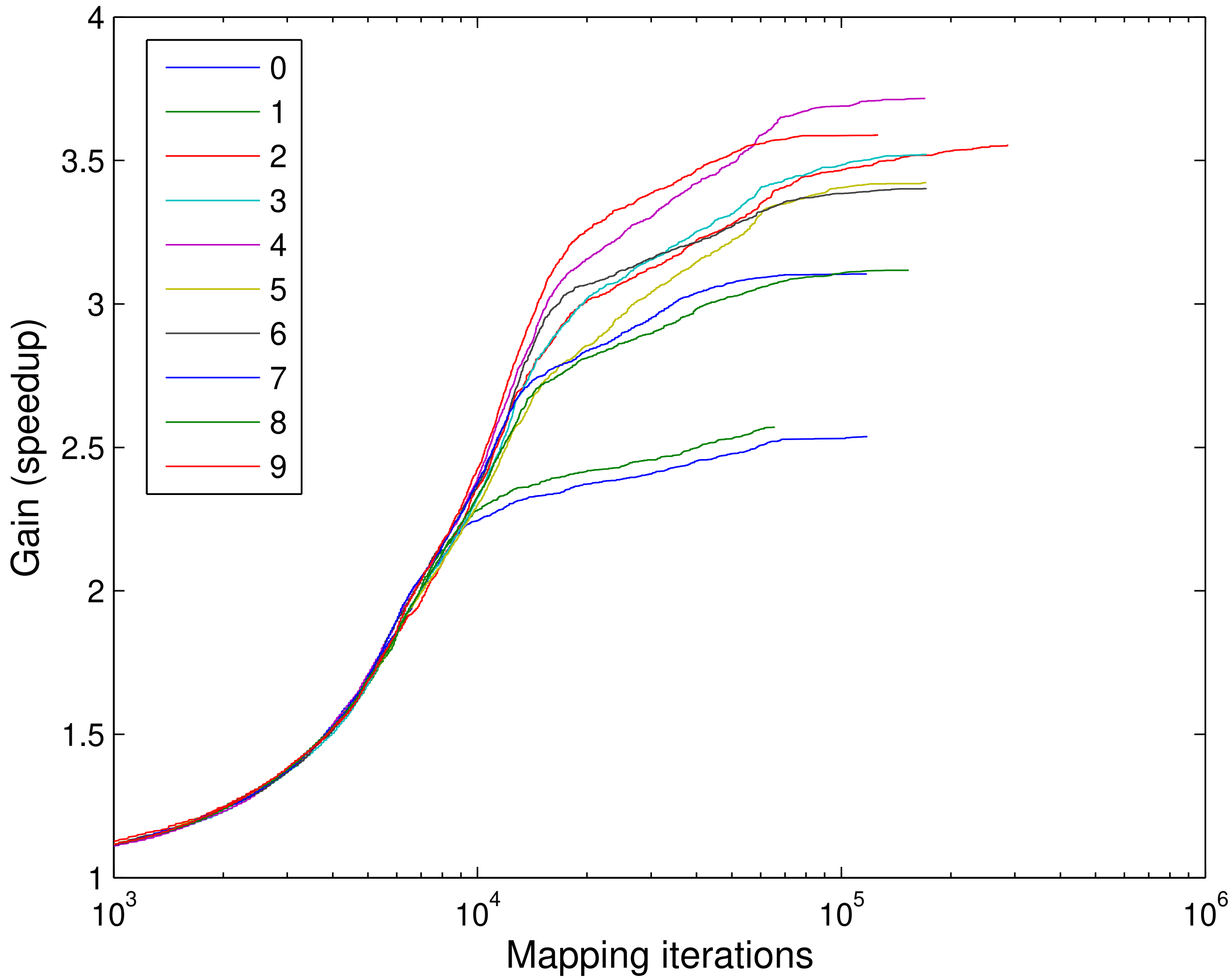
# Rounds and mapping iterations for OSM

| PEs | rounds (min, avg, max) | Thousands of iterations (min, avg, max) |
|---|---|---|
| 2 | 271, 380, 611 | 34.1, 37.2, 73.6 |
| 4 | 239, 469, 899 | 80.6, 115.4, 259.1 |
| 8 | 199, 428, 1099 | 57.1, 88.8, 293.9 |

# Best gain divided by the number of iterations

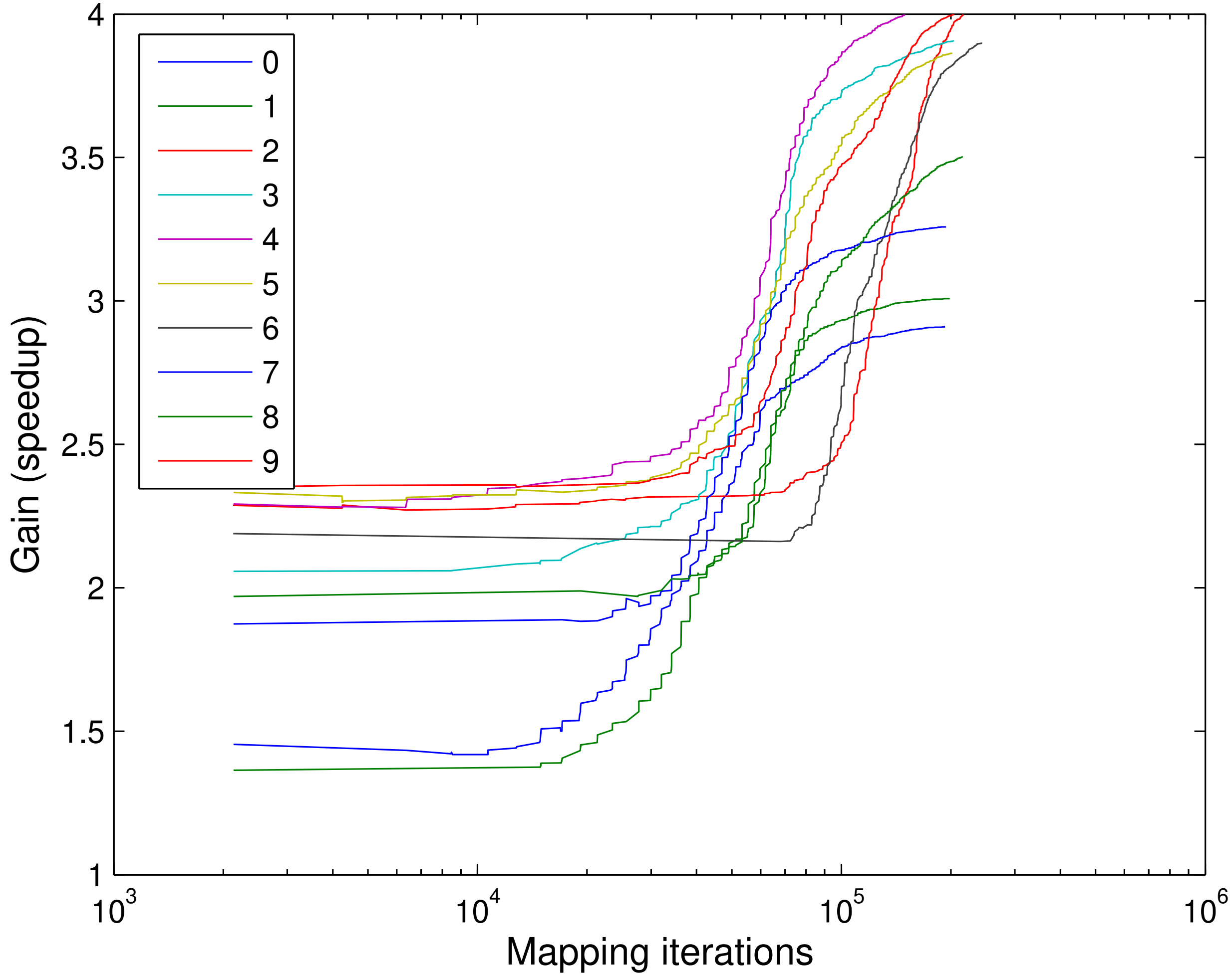


# OSM progress plotted for each graph



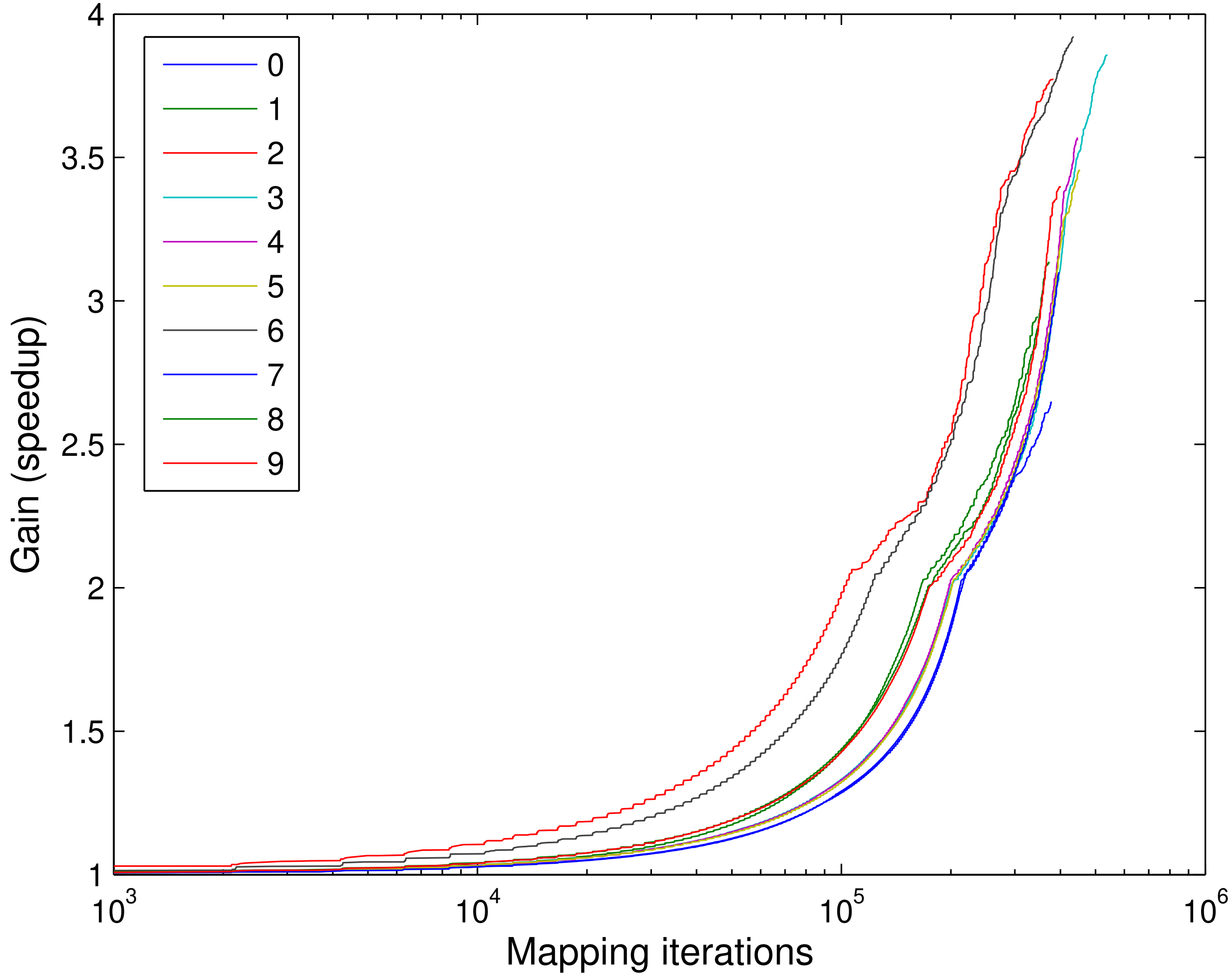

TAMPERE UNIVERSITY OF T

# SA+AT progress plotted for each graph




TAMPERE UNIVERSITY OF T

# GM progress plotted for each graph




TAMPERE UNIVERSITY OF T