Colorado State University

Memory Allocations for Tiled Uniform Dependence Programs

Tomofumi Yuki and Sanjay Rajopadhye

Parametric Tiling

- Series of advances
  - Perfect loop nests [Renganarayanan2007]
  - Imperfectly nested loops [Hartono2009, Kim2009]
  - Parallelization [Hartono2010, Kim2010]
- Key idea:
  - Step out of the polyhedral model
    - Parametric tiling is not affine
  - Use syntactic manipulations

Memory Allocations

- Series of polyhedral approaches
  - Affine Projections [Wilde & Rajopadhye 1996]
  - Pseudo-Projections [Lefebvre & Feautrier 1998]
  - Dimension-wise "optimal" [Quilleré & Rajopadhye 2000]
  - Lattice-based [Darte et al. 2005]
- Cannot be used for parametric tiles
  - Can be used to allocate *per tile* [Guelton et al. 2011]
- Difficult to combine parametric tiling with memory-reallocation

This paper

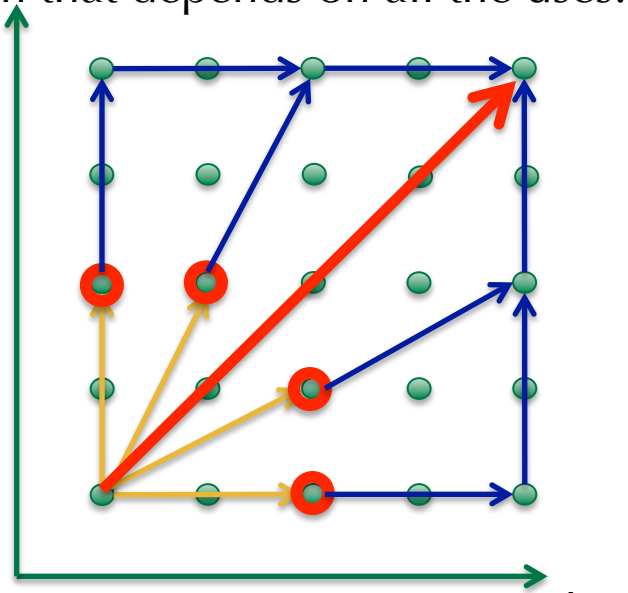
- Find allocations valid for a set of schedules
  - Tiled execution by **any** tile size
  - Based on Occupancy Vectors [Strout et al. 1998]
    - Restrict the universe to tiled execution
    - Quasi-Universal Occupancy Vectors
    - More compact allocations than UOV
- Analytically find the shortest Quasi-UOV
- UOV-guided index-set splitting
  - Separate boundaries to reduce memory usage

Outline

- Introduction
- Universal Occupancy Vectors (review)
- Lengths of UOVs
- Overview of the proposed flow
- Finding the shortest QUOV
- UOV-guided Index-set Splitting
- Related Work
- Conclusions

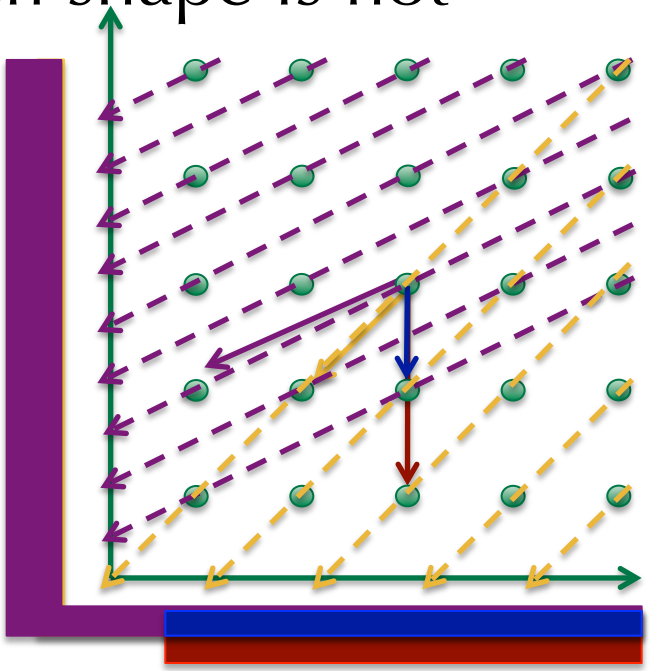
Universal Occupancy Vectors

- Find a valid allocation for any legal schedules
- Occupancy vector: `ov`
  - Value produced at `z` is dead by `z+ov`
- Assumptions
  - Same dependence pattern
  - Single statement
  - Legal schedule can even be from run-time scheduler



Lengths of UOVs

- Shorter ≠ Better
  - The shape of iteration space has influence
- A good "rule of thumb" when shape is not known
- Increase in Manhattan distance usually leads increase in memory usage



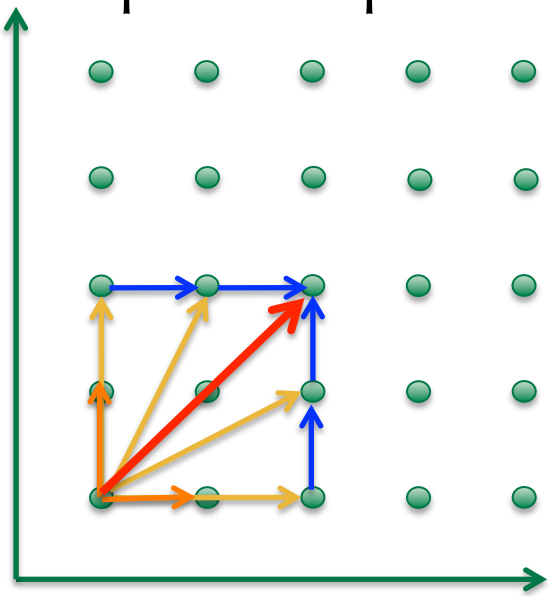
Proposed Flow

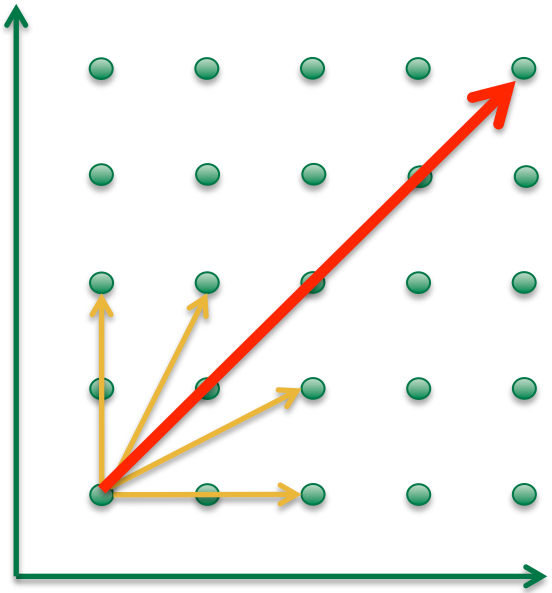
- Input: Polyhedral representation of a program
  - no memory-based dependences
- Make scheduling choices
  - The result should be (partially) tilable
- Apply schedules as affine transforms
  - Lex. scan of the space now reflects schedule
- Apply UOV-based index-set splitting
- Apply QUOV-based allocation

UOV for Tilable Space

- We know that the iteration space will be tiled
  - Dependences are always in the first orthant
  - Certain order is always imposed

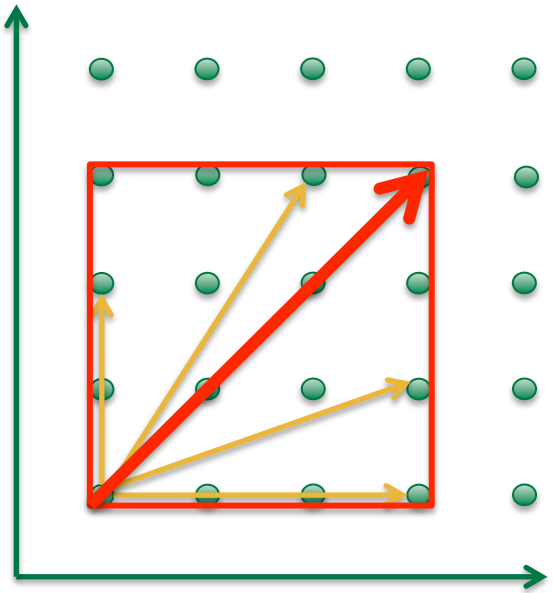
➔ Implicit dependences





Finding the shortest QUOV

- 1. Create a bounding hyper-rectangle
  - Smallest that contains all dependences
- 2. The diagonal is the shortest UOV
- Intuition
  - No dependence goes "backwards"
  - Property of tilable space



Outline

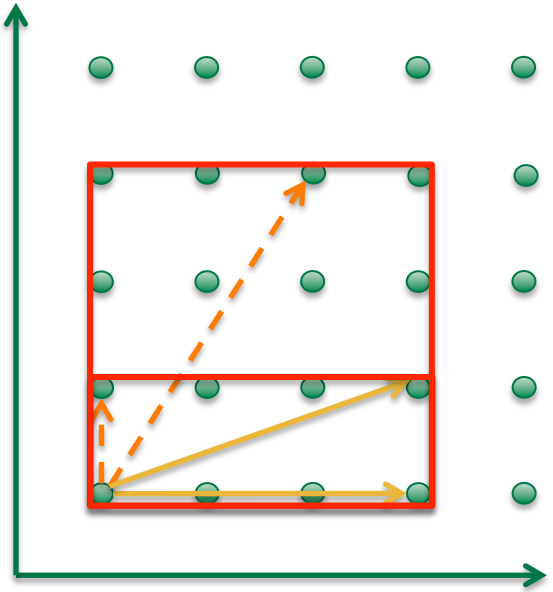
- Introduction
- Universal Occupancy Vectors (review)
- Lengths of UOVs
- Overview of the proposed flow
- Finding the shortest QUOV
- UOV-guided Index-set Splitting
- Related Work
- Conclusions

Dependences at Boundaries

- Many boundary conditions in polyhedral representation of programs
  - e.g., Gauss Seidel 2D (from polybench)
    - Single C statement, 10+ boundary cases
  - May negatively influence storage mapping
    - With per-statement projective allocations
  - Different life-times at boundaries
    - May be longer than the main body
- Allocating separately may also be inefficient

UOV-Based Index-Set Splitting

- "Smart" choice of boundaries to separate out
  - Those that influence the shortest QUOV
- Example:
  - Dashed dependences = boundary dependences
  - Removing one has no effect
  - Removing the other shrinks the bounding hyper-rect.

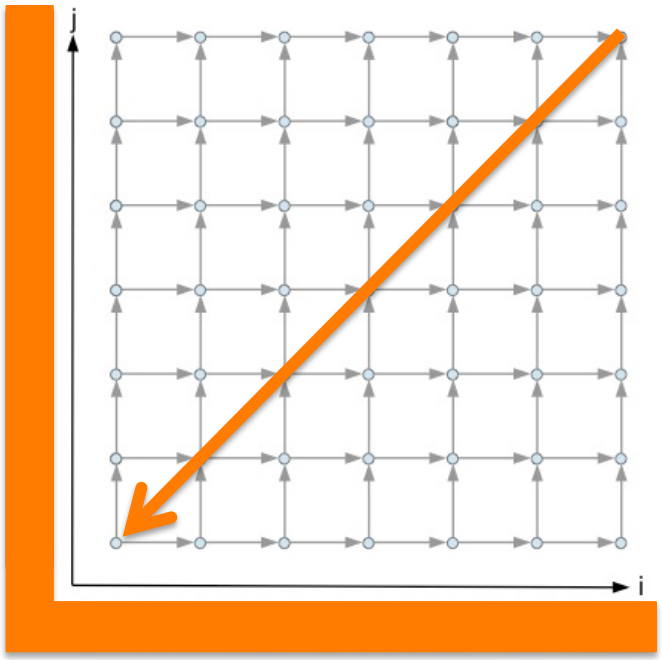


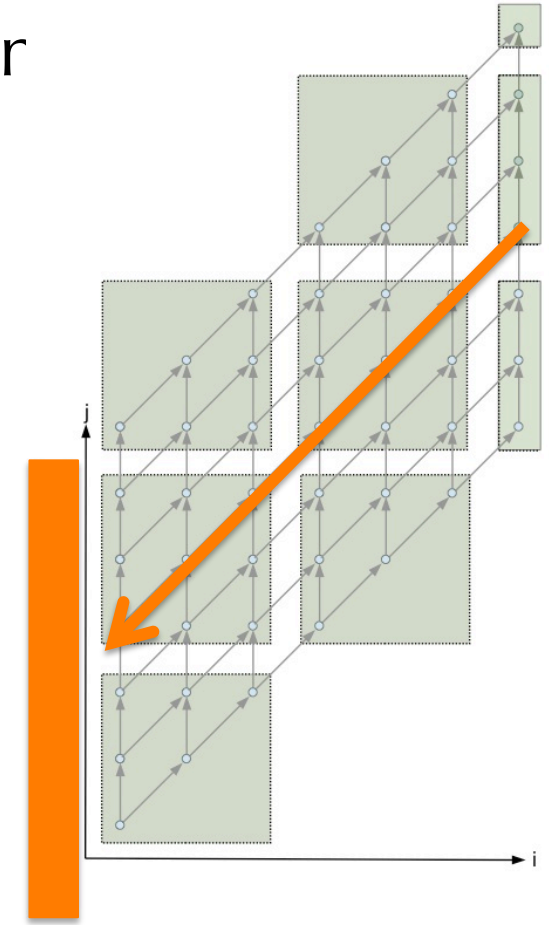
Related Work

- Affine Occupancy Vectors [Thies et al. 2001]
  - Restrict the universe to affine schedules
- Comparison with schedule-dependent methods
  - Schedule-dependent methods are at least as good as UOV or QUOV based approaches
  - UOV based methods may not be as inefficient as one might think
    - Provided O(d-1) data is required for d dimensional space
    - UOV-based methods are *single* projection

Example

- Smith-Waterman (-like) deper





Summary and Conclusion

- We "expand" the concept of UOV to a smaller universe: tiled execution
- We use properties in such universe to find:
  - More compact allocations
  - Shortest QUOVs
  - Profitable index-set splitting
- Possible approach for parametrically tiled programs

Acknowledgements

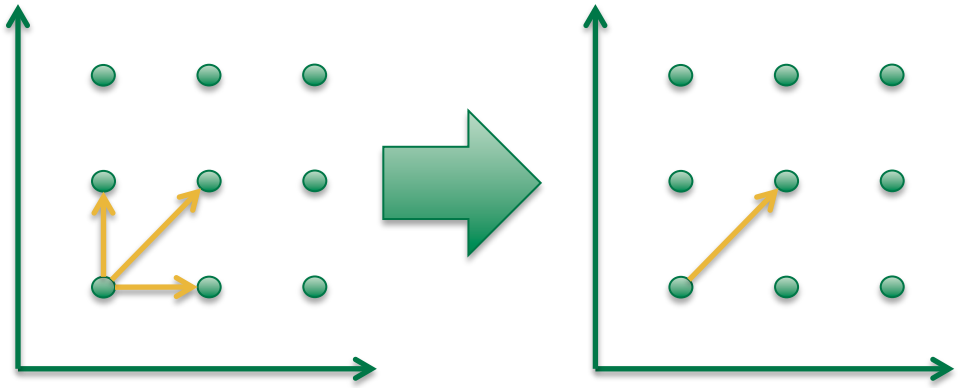
- Michelle Strout
  - For discussion and feedback
- IMPACT PC and Chairs
  - Our paper is in a much better shape after revisions

Extensions to Multi-Statement

- Schedule-Independent mapping is for programs with single statement
  - We reduce the universality to tiled execution
  - Multi-statement programs can be handled
- Intuition:
  - When tiling a loop nest, the same affine transform (schedule) is applied to all statements
  - Dependences remain the same

Dependence Subsumption

- Some dependences may be excluded when considering UOVs and QUOVs
- A dependence `f` **subsumes** a set of dependences *`I`* if `f` can be expressed transitively by dependences in *`I`*



Valid UOV for the left is also valid for the right.