CITED BY *9*

Space-Time Trade-Off Optimization for a Class of Electronic Structure Calculations

Daniel Cociorva[1] Gerald Baumgartner[1] Chi-Chung Lam[1] P. Sadayappan[1]
J. Ramanujam[2] Marcel Nooijen[3] David E. Bernholdt[4] Robert Harrison[5]

[1] Dept. of Computer and Information Science
The Ohio State University
{cociorva,gb,clam,saday}
@cis.ohio-state.edu

[2] Dept. of Electrical and Computer Engineering
Louisiana State University
jxr@ece.lsu.edu

[3] Department of Chemistry
Princeton University
Nooijen@Princeton.edu

[4] Oak Ridge National Laboratory
bernholdtde@ornl.gov

[5] Pacific Northwest National Laboratory
Robert.Harrison@pnl.gov

読んだ人: みよしたけふみ

100 to 1000TB

$$S_{abij} = \sum_{cdefkl} A_{acik} \times B_{befl} \times C_{dfjk} \times D_{cdel}$$

$$S_{abij} = \sum_{ck} \left(\sum_{df} \left(\sum_{el} B_{befl} \times D_{cdel} \right) \times C_{dfjk} \right) \times A_{acik}$$

$$\begin{aligned} T1_{bcdf} &= \sum_{el} B_{befl} \times D_{cdel} \\ T2_{bcjk} &= \sum_{df} T1_{bcdf} \times C_{dfjk} \\ S_{abij} &= \sum_{ck} T2_{bcjk} \times A_{acik} \end{aligned}$$

(a) Formula sequence

```
T1=0; T2=0; S=0
for b, c, d, e, f, l
[ T1bcdf += Befl Dcdel
for b, c, d, f, j, k
[ T2bcjk += T1bcdf Cdfjk
for a, b, c, i, j, k
[ Sabij += T2bcjk Aacik
```

(b) Direct implementation (unfused code)

```
S = 0
for b, c
[ T1f = 0; T2f = 0
  for d, f
  [ for e, l
    [ T1f += Befl Dcdel
    for j, k
    [ T2fjk += T1f Cdfjk
  for a, i, j, k
  [ Sabij += T2fjk Aacik
```

(c) Memory-reduced implementation (fused)

Figure 1: Example illustrating use of loop fusion for memory reduction.

Optimization System

- Algebraic Transformations
- Memory Minimization
- Space-Time Transformation
- Data Locality Optimization
- Data Distribution and Partitioning

Fusion Graph

can be used to facilitate enumeration of all possible compatible fusion configurations for a given computation tree.

The potential for fusion of a common loop among a producer-consumer pair of loop nests is indicated in the fusion graph through a dashed edge connecting the corresponding vertices.

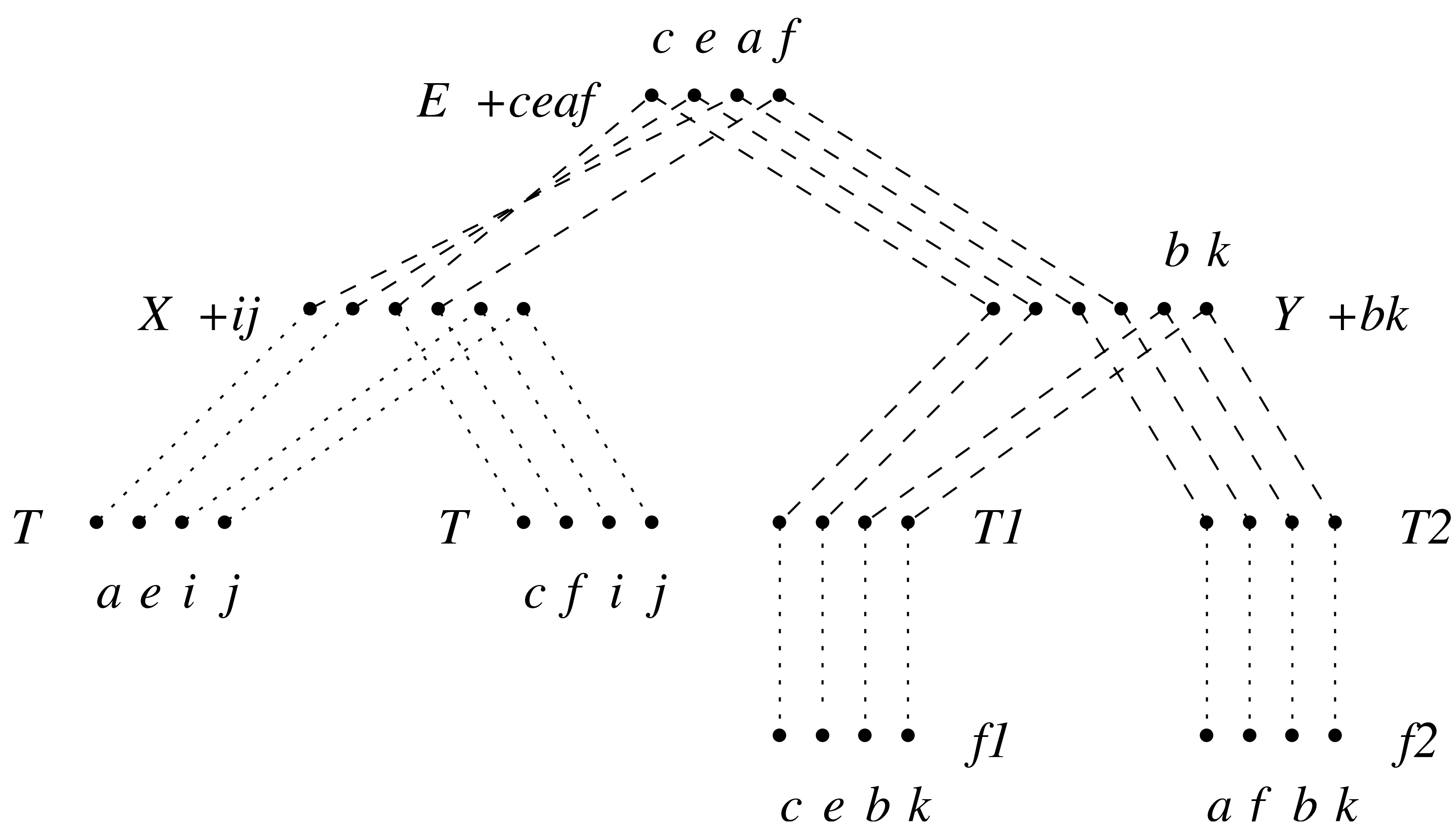


Figure 5: Fusion graph for unfused operation-minimal form of loop in Figure 2.

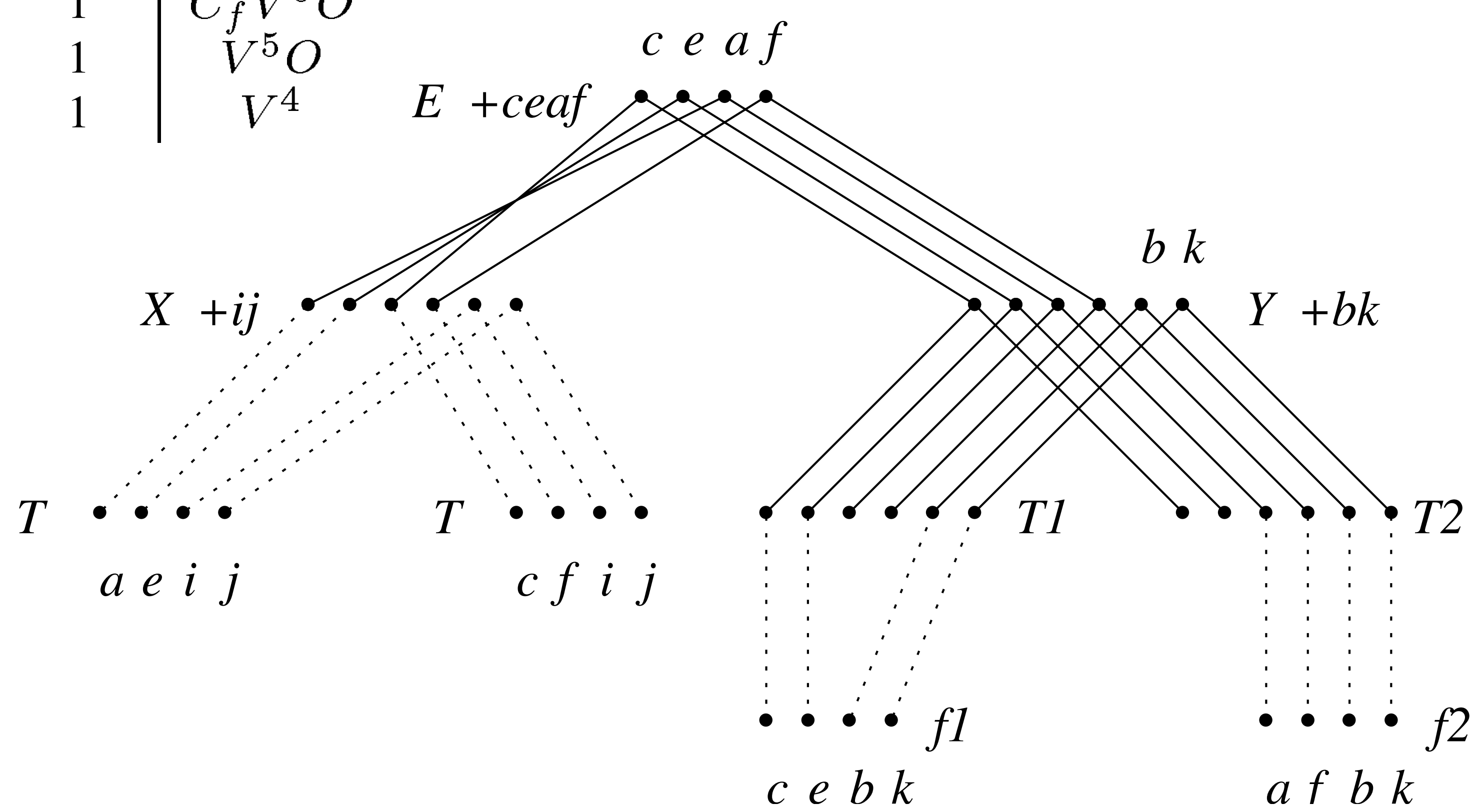
Example (1)

```
for a, e, c, f
[ for i, j
[ [ X_aecf += T_ijae T_ijcf
for a, f
[ for c, e, b, k
[ [ T1_cebk = f_1(c,e,b,k)
for c, e
[ for a, f, b, k
[ [ T2_afbk = f_2(a,f,b,k)
for c, e, a, f
[ for b, k
[ [ Y_ceaf += T1_cebk T2_afbk
for c, e, a, f
[ E += X_aecf Y_ceaf
```

⇒

```
for a, e, c, f
[ for i, j
[ [ X += T_ijae T_ijcf
[ for b, k
[ [ T1 = f_1(c,e,b,k)
[ [ T2 = f_2(a,f,b,k)
[ [ Y += T1 T2
[ E += X Y
```

array	space	time
X	1	V^4O^2
T1	1	C_fV^5O
T2	1	C_fV^5O
Y	1	V^5O
E	1	V^4

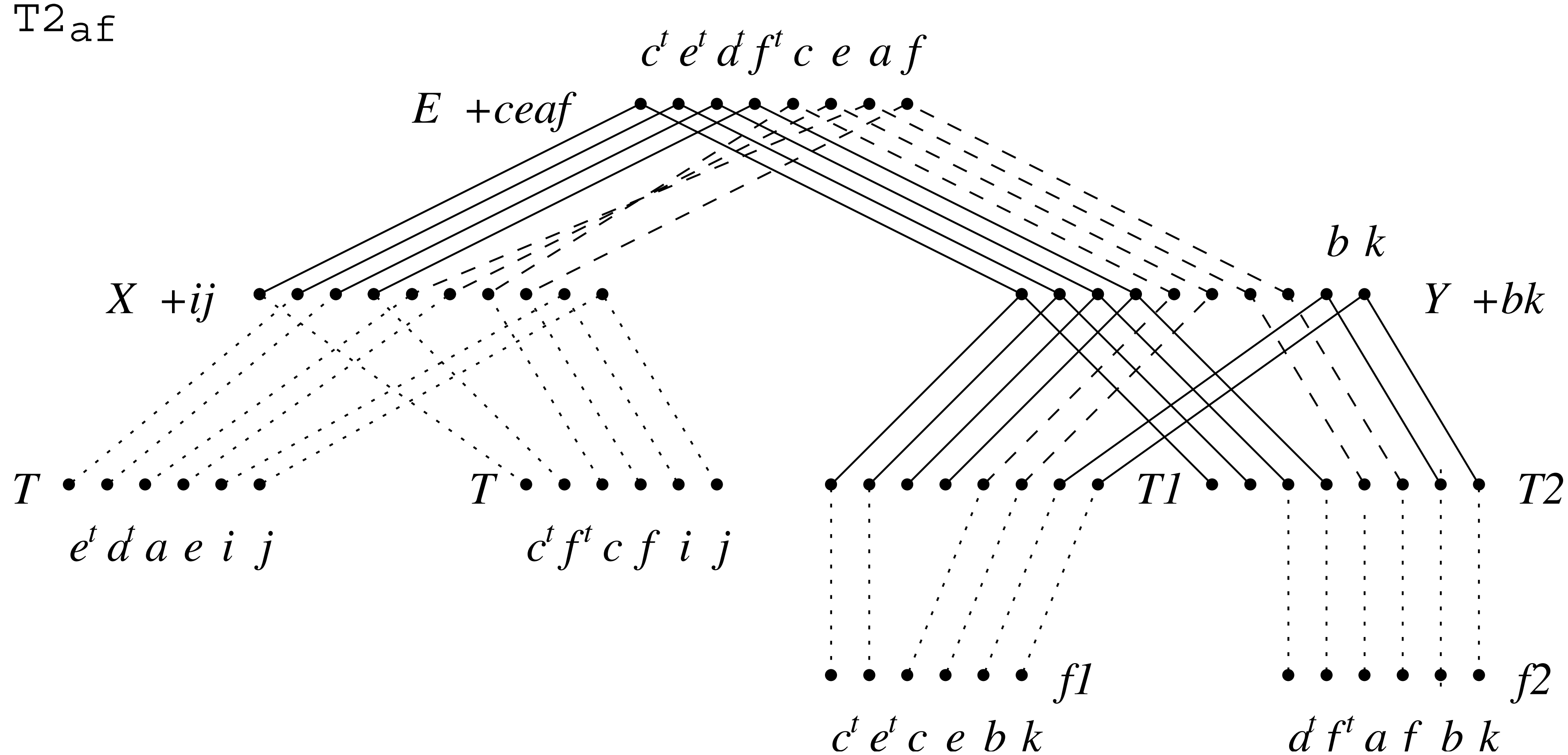


(a) Fully fused computation from Fig. 3.

Example (2)

```
for a^t, e^t, c^t, f^t
  for a, e, c, f
    for i, j
      X_aecf += T_ijae T_ijcf
  for b, k
    for c, e
      T1_ce = f_1(c,e,b,k)
    for a, f
      T2_af = f_2(a,f,b,k)
    for c, e, a, f
      Y_ceaf += T1_ce T2_af
  for c, e, a, f
    E += X_aecf Y_ceaf
```

array	space	time
X	B^4	V^4O^2
T1	B^2	$C_f V^5 O/B^2$
T2	B^2	$C_f V^5 O/B^2$
Y	B^4	V^5O
E	1	V^4



(b) Partially fused computation from Fig. 4.

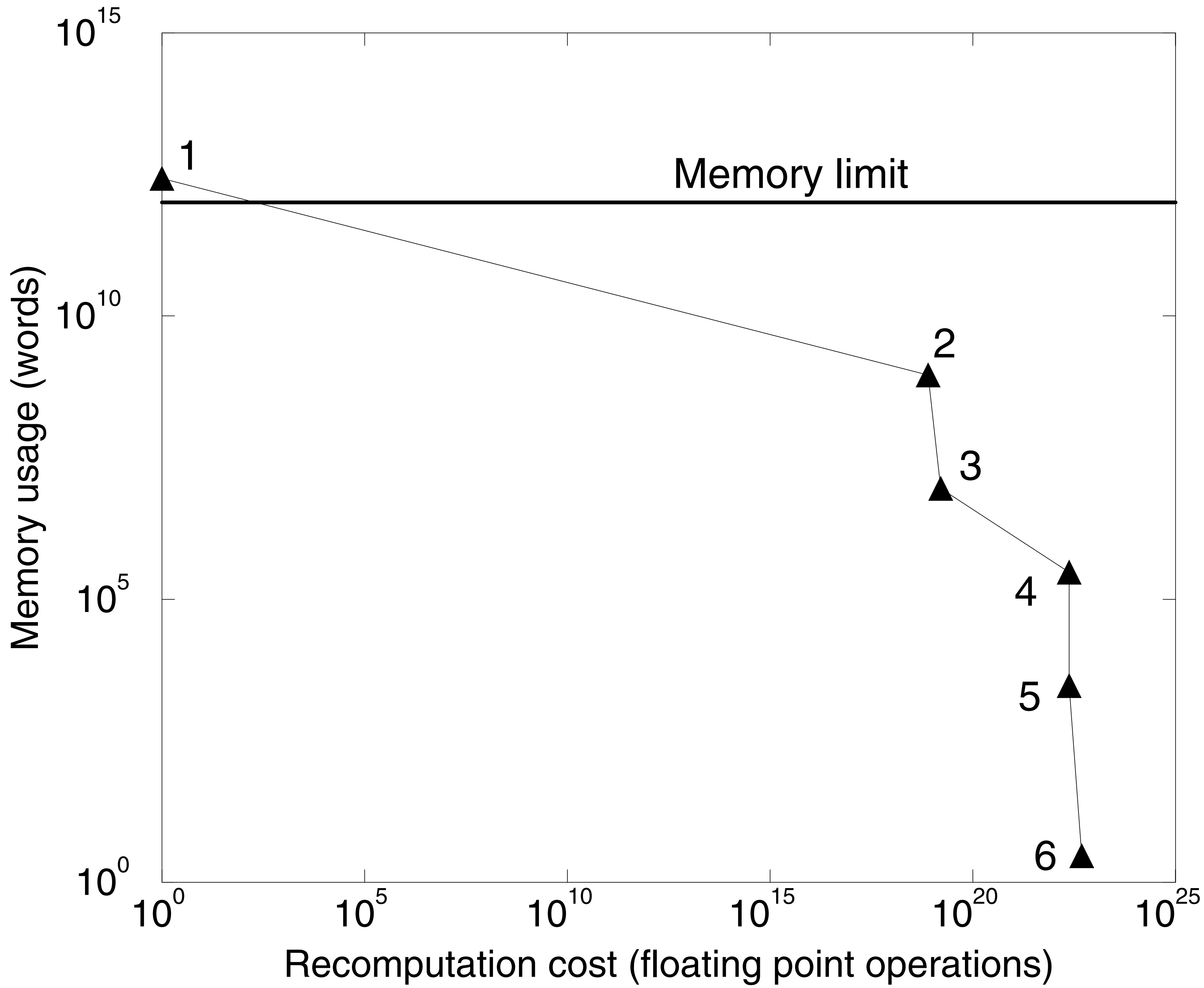
Space-Time Tradeoff Exploration

- Search among all possible ways of introducing redundant loop indices in the fusion graph to reduce memory requirements, and determine the optimal set of lower dimensional intermediate arrays for various total memory limits. In this step, the use of tiling for partial reduction of array extents is not considered. However, among all possible combinations of lower dimensional arrays for intermediates, the combination that minimizes recomputation cost is determined, for a specified memory limit. The range from zero to the actual memory limit is split into subranges within which the optimal combination of lower dimensional arrays remains the same.
- Because the first step only considers complete fusion of loops, each array dimension is either fully eliminated or left intact, i.e. partial reduction of array extents is not performed. The objective of the second step is to allow for such arrays. Starting from each of the optimal combinations of lower dimensional intermediate arrays derived in the first step, possible ways of using tiling to partially expand arrays along previously compressed dimensions are explored. The goal is to further reduce recomputation cost by partially expanding arrays to fully utilize the available memory

Space-Time optimization

- Dimension Reduction for Intermediate Arrays
  - search among all possible combination
  - memory and recomputation costs
- Partial Expansion of Reduced Intermediates
  - resort to array expansion
  - for determining the best choice for array expansion costs

Result



Memory limit
Memory usage (words)
Recomputation cost (floating point operations)
1
2
3
4
5
6
10^{15}
10^{10}
10^{5}
10^{0}
10^{0}
10^{5}
10^{10}
10^{15}
10^{20}
10^{25}