Convex clustering
(work in progress)

B. Gaujal, G. Huard, J. Pecero, E. Thierry and D. Trystram

Laboratoire ID-IMAG

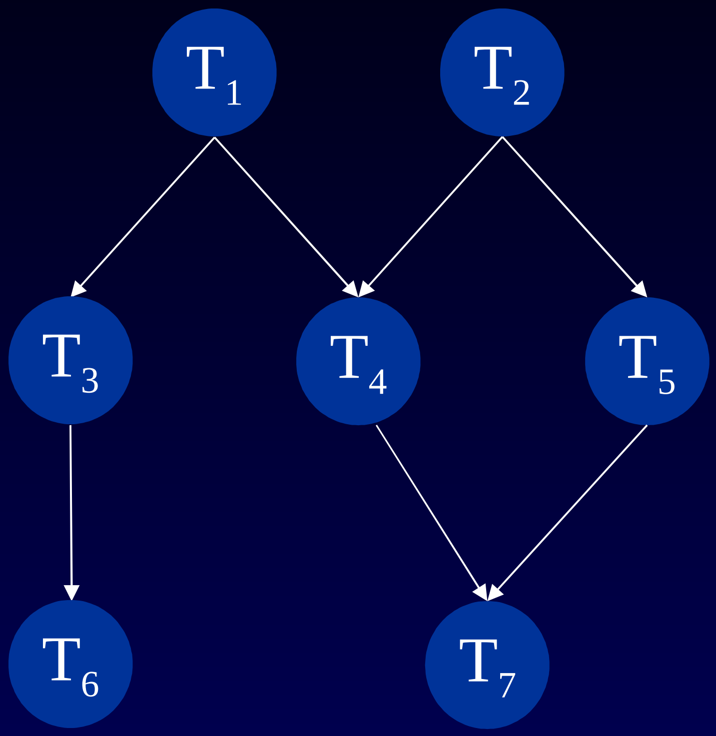
Denis.Trystram@imag.fr

Guillaume.Huard@imag.fr

Context and Motivation

- High-performance computing on distributed memory architectures (PC clusters and grid infrastructures).
  - High latency of interconnection network.
- The objective is to schedule a parallel application:
  - Determining where and when to execute the tasks
  - Minimize the makespan (denoted by ω)
- Focus: taking into account large communication costs into the scheduling decision is a key point to reach high performance.

Classical application model: precedence task graph



- Vertices:

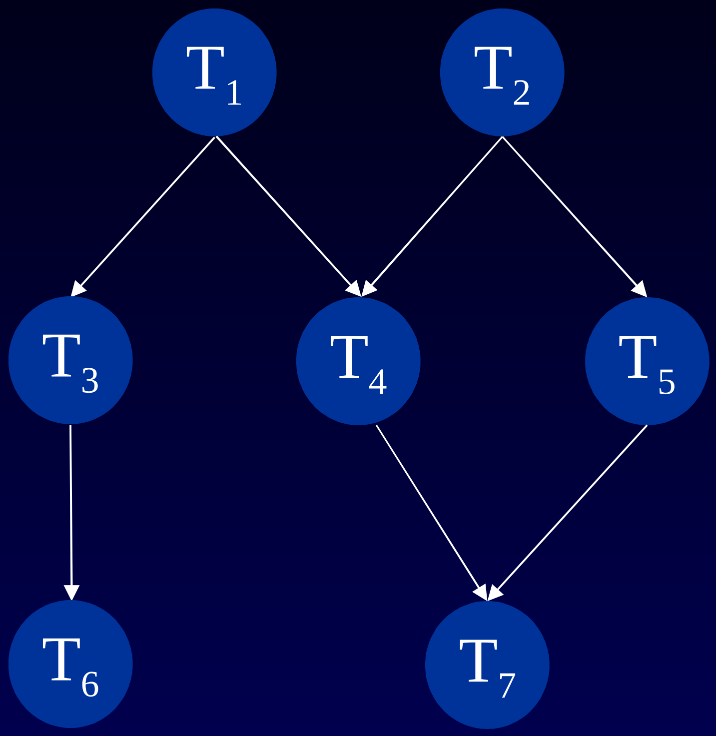
  computation tasks

- Edges:

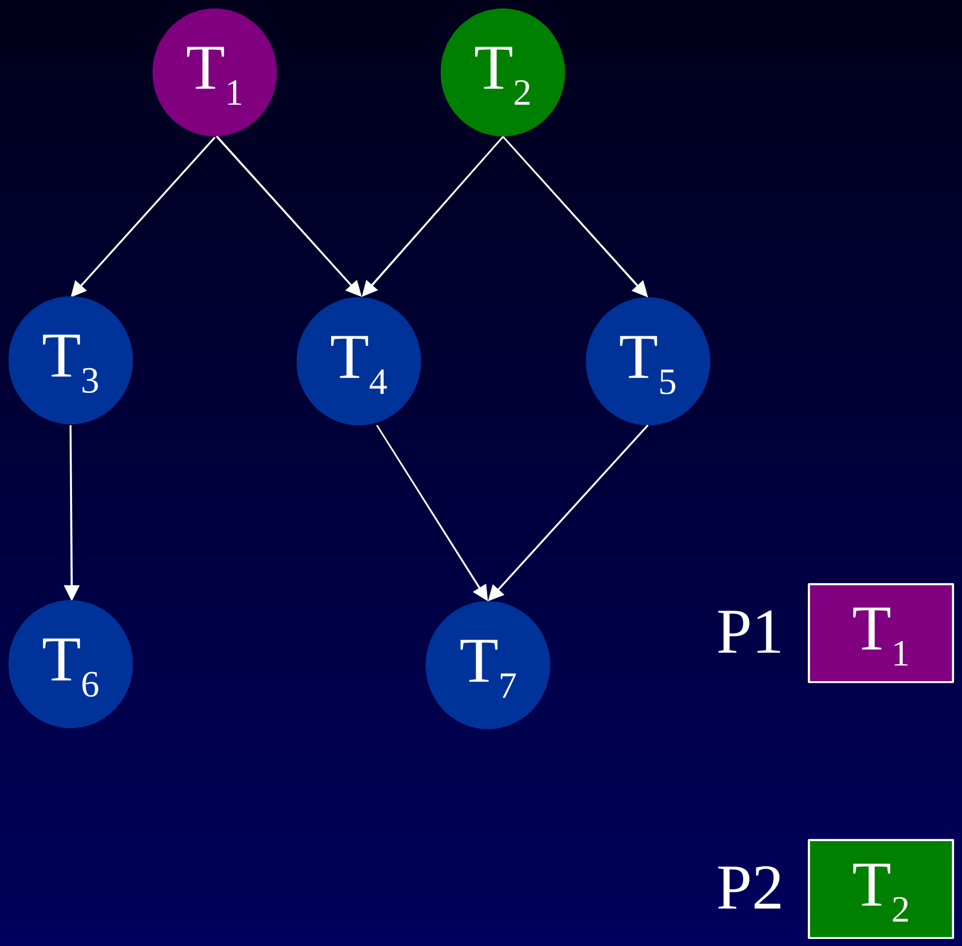
  data dependencies between tasks.

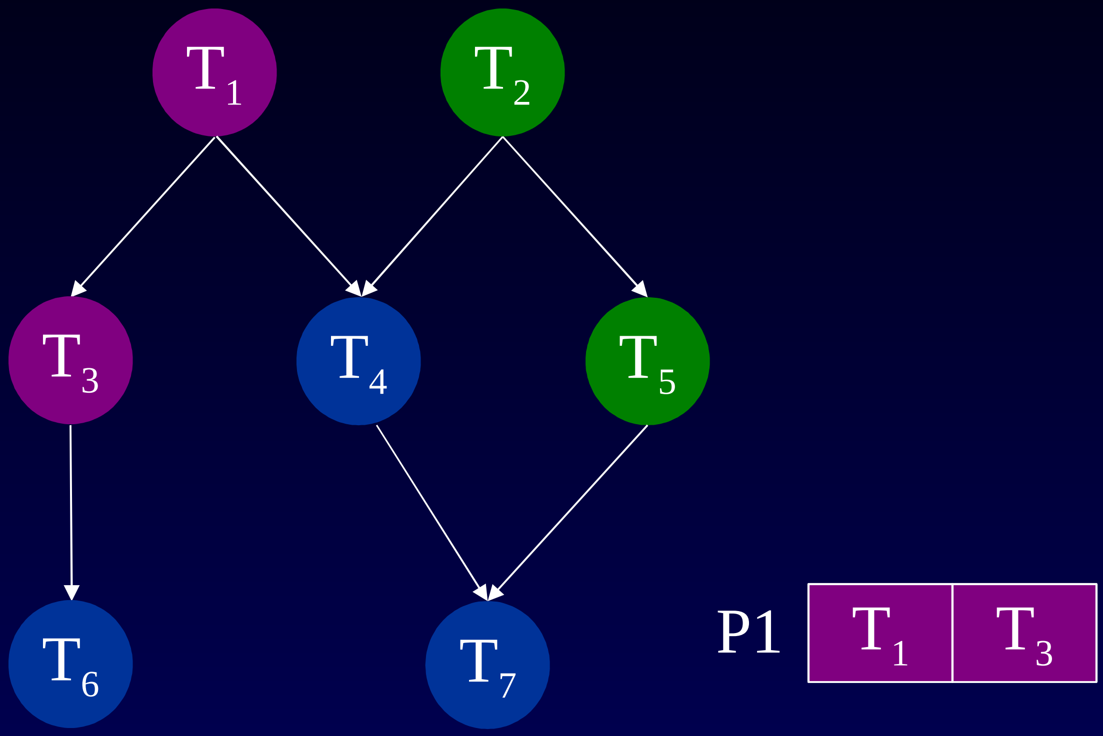
The delay model

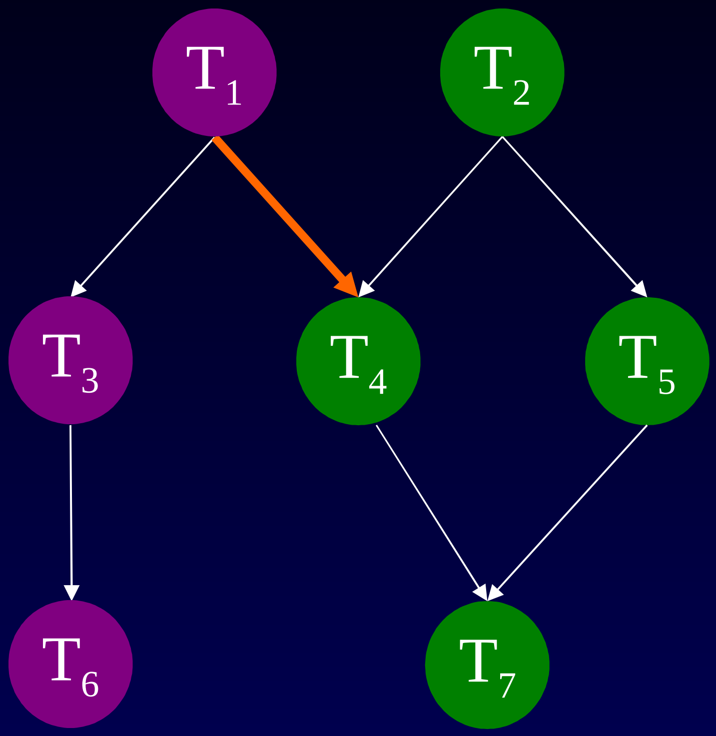
[Rayward-Smith 86, Papadimitriou and Yannakakis 90]



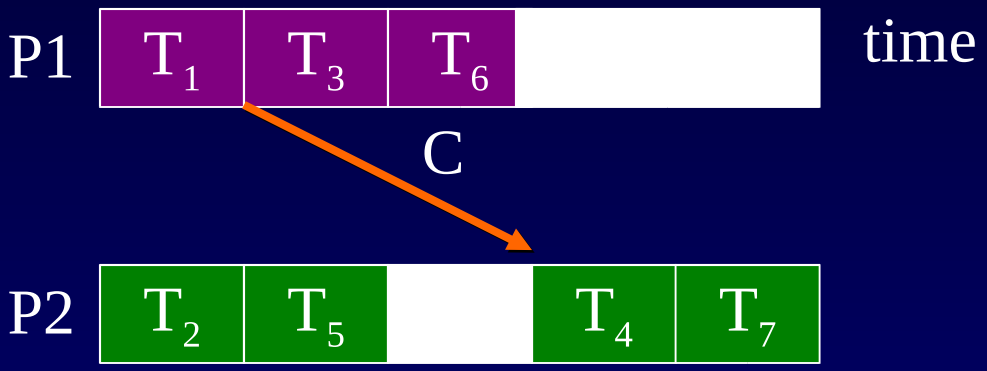
- Unit execution time for tasks
- Communications at a fixed cost *c*


T_1
T_2
T_3
T_4
T_5
T_6
T_7
P1
T_1
P2
T_2


T1
T2
T3
T4
T5
T6
T7
P1
T1
T3
P2
T2
T5



- Simple model :
  - no overhead
  - no contention
  - total overlap



Recall of the basic scheduling problem

Instance: Precedence task graph and a delay C

Problem: Choose for each task a location and a date

Objective: minimize the makespan

NP-hard problem even for simple cases.

No constant guaranty for large communication delays!

Recall of the basic scheduling problem

Large number of heuristics, three main families:

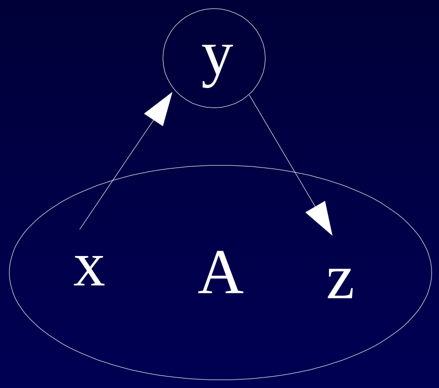
- Extention of list algorithms:
  ETF (Earliest task first) [Hwang and al.89]
- Locations assignment based on critical path:
  DSC (Dominant sequence clustering)
  [Gerasoulis and Yang 94]
- Graph decomposition:
  CLANS [McCreary and al. 89]

Convex clustering

- Idea:

  assign tasks to locations in convex groups
- Convexity:

  A cluster *A* is convex iff $\forall x, z \in A, x \rightarrow y \wedge y \rightarrow z \Rightarrow y \in A$

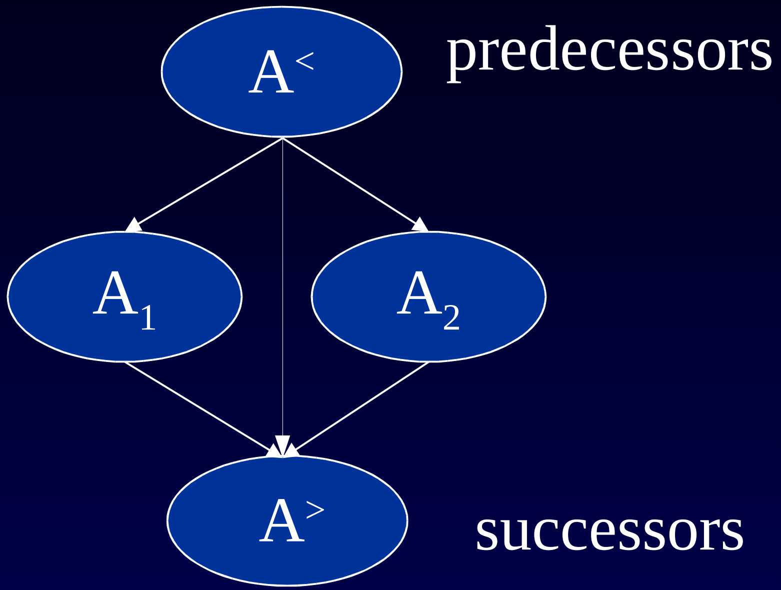


Characteristics

- Advantages:
  - Schedules based on convex clusters are 2-dominant
    [Trystram and Lepere 2000]
  - The resulting graph is acyclic and:
    - clustering makes the grain coarser
    - classical guaranteed algorithms can be used thereafter
- Related approach:
  - CLANS [McCreary and al. 89] are special case of convex clusters

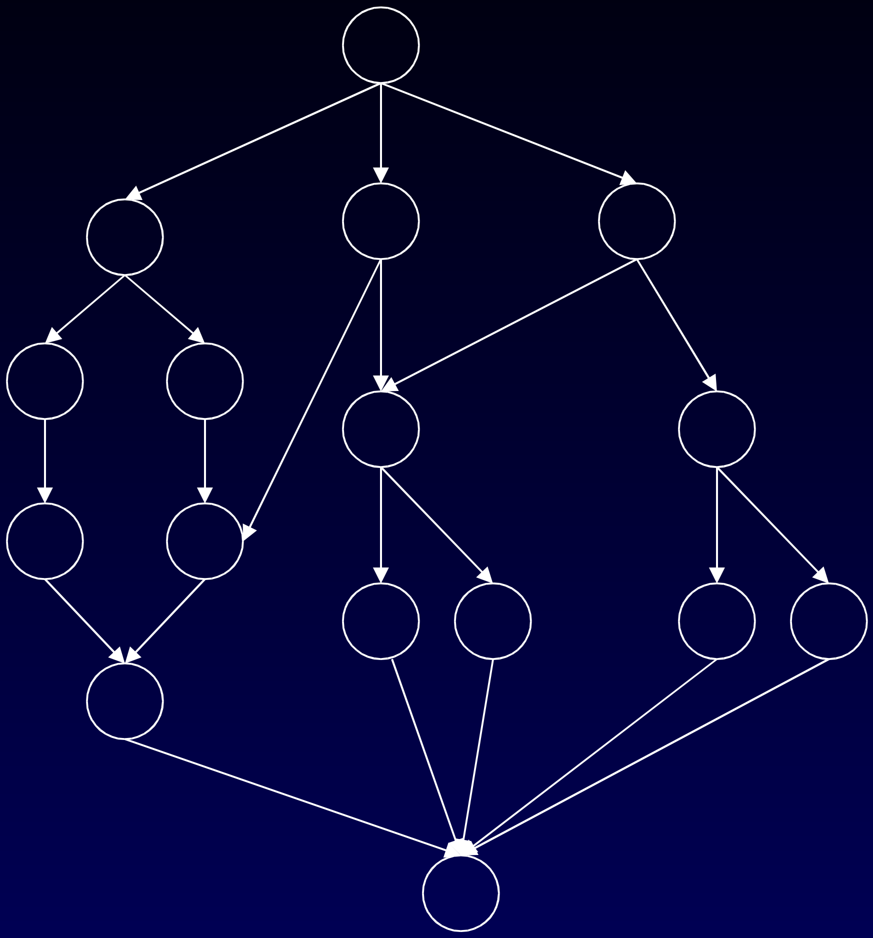
Recursive approach of the problem

- Find in graph G=(V,E) two independent sets of tasks A_1 and A_2.

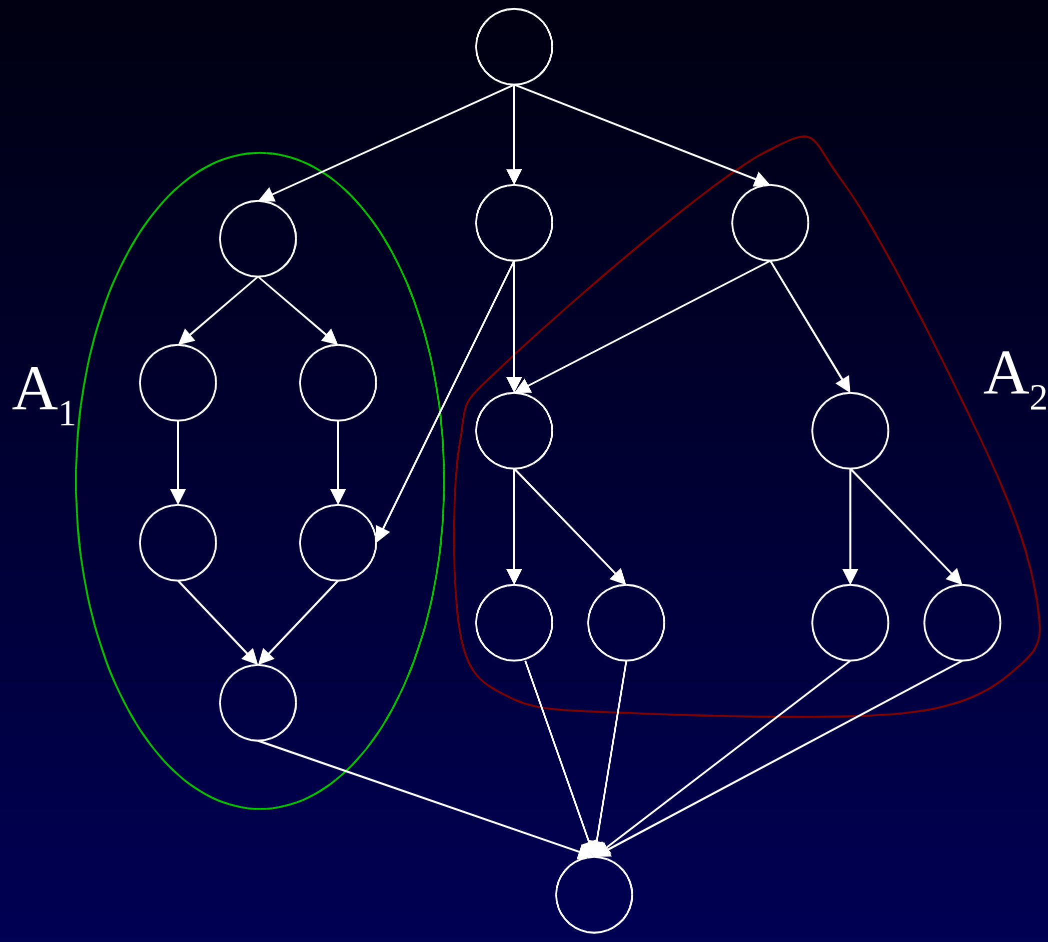


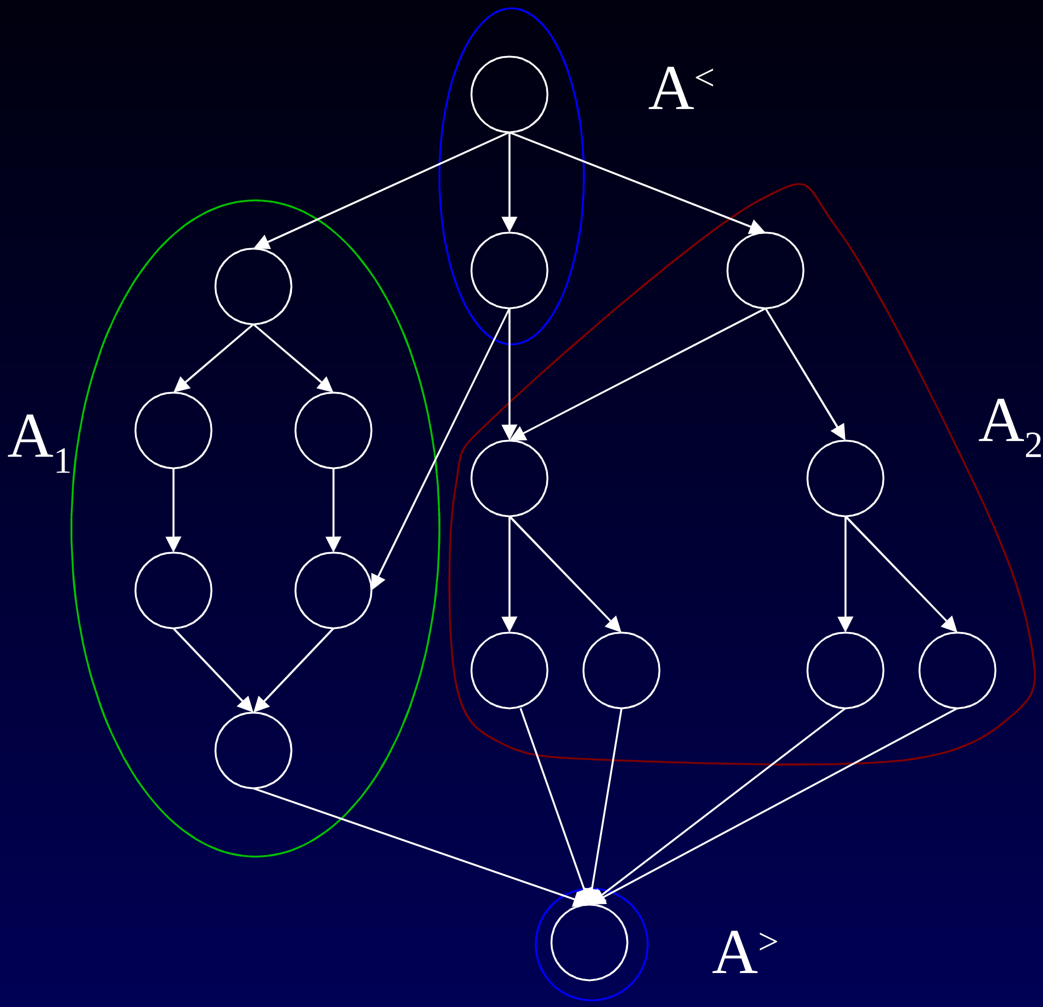
- Split recursively A_1, $A_{2,}$ $A^>$ and $A^<$ if such a splitting allows to decrease the longest path in regard to a sequential execution.

Illustration

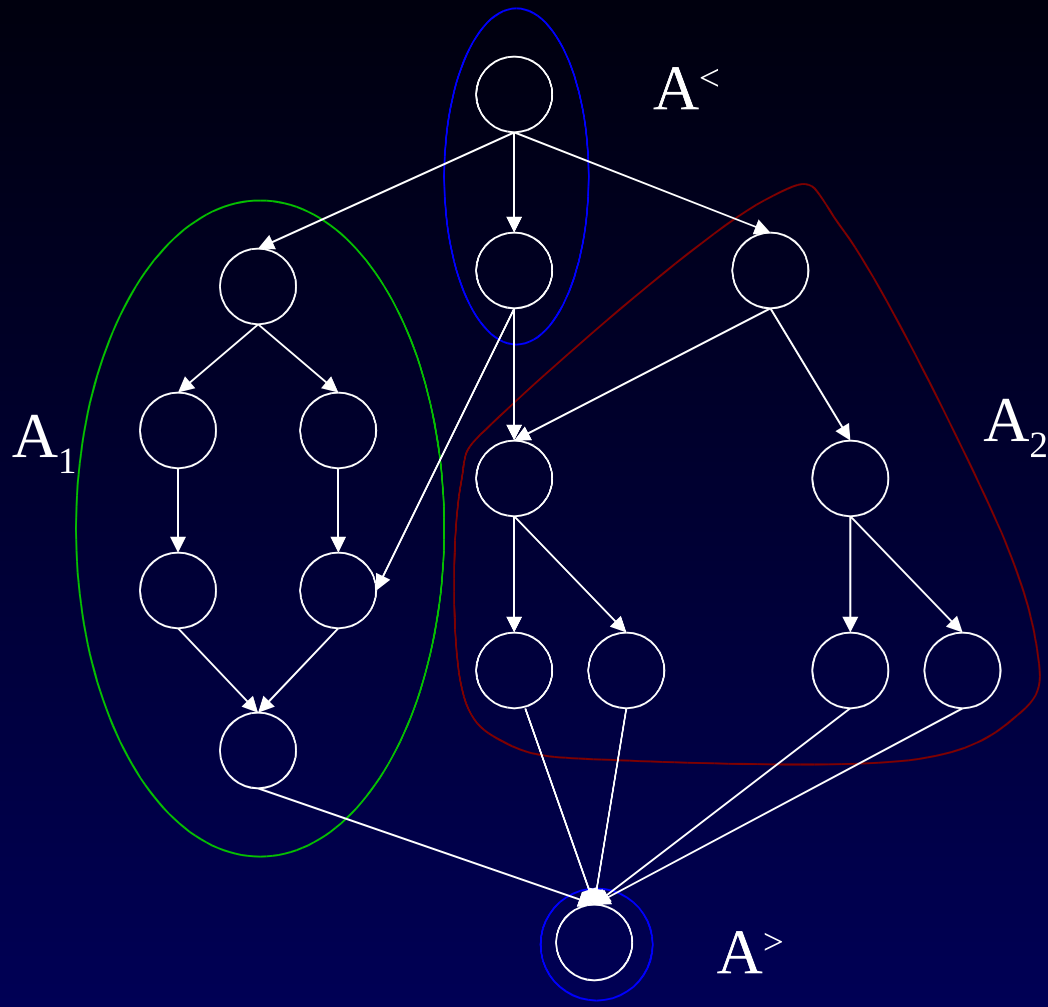


Graph with 16 tasks and
communication delay C = 2.

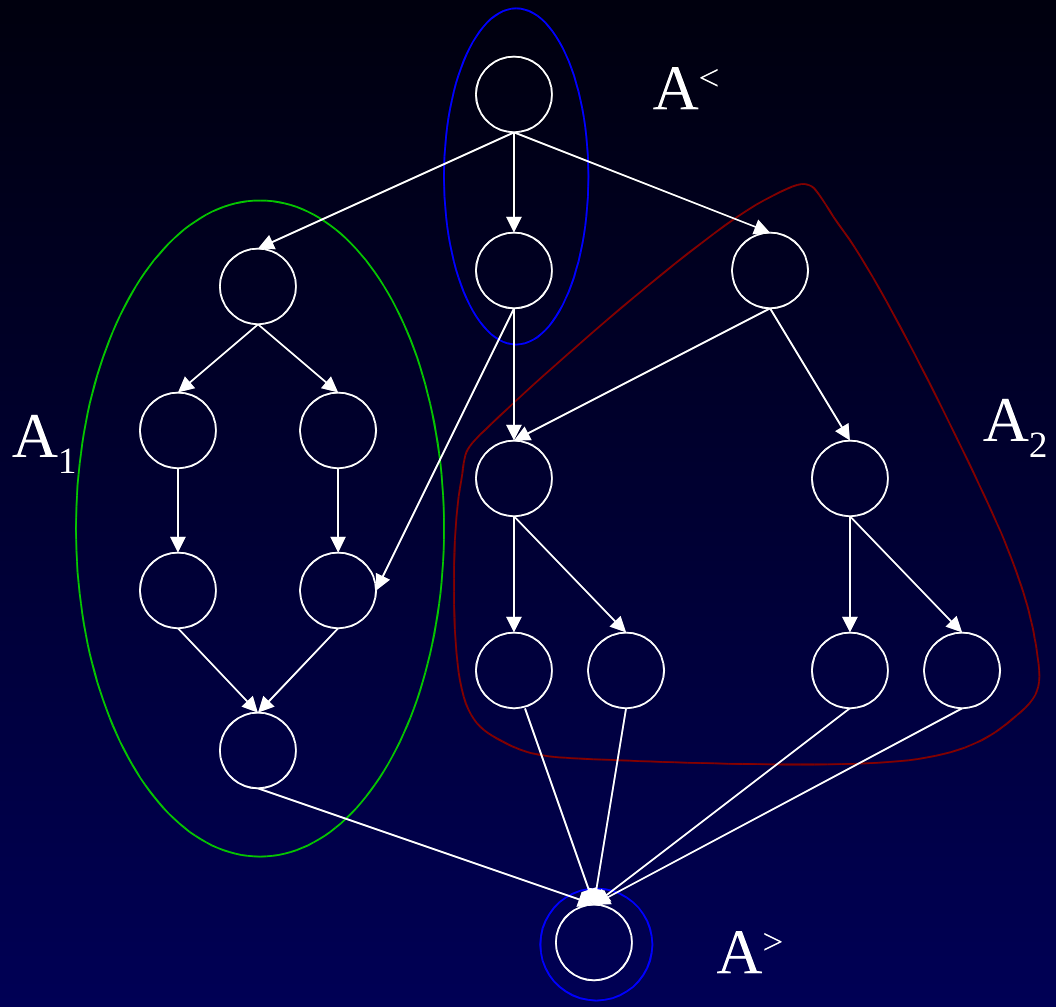

A_1
A_2



A<
A1
A2
A>



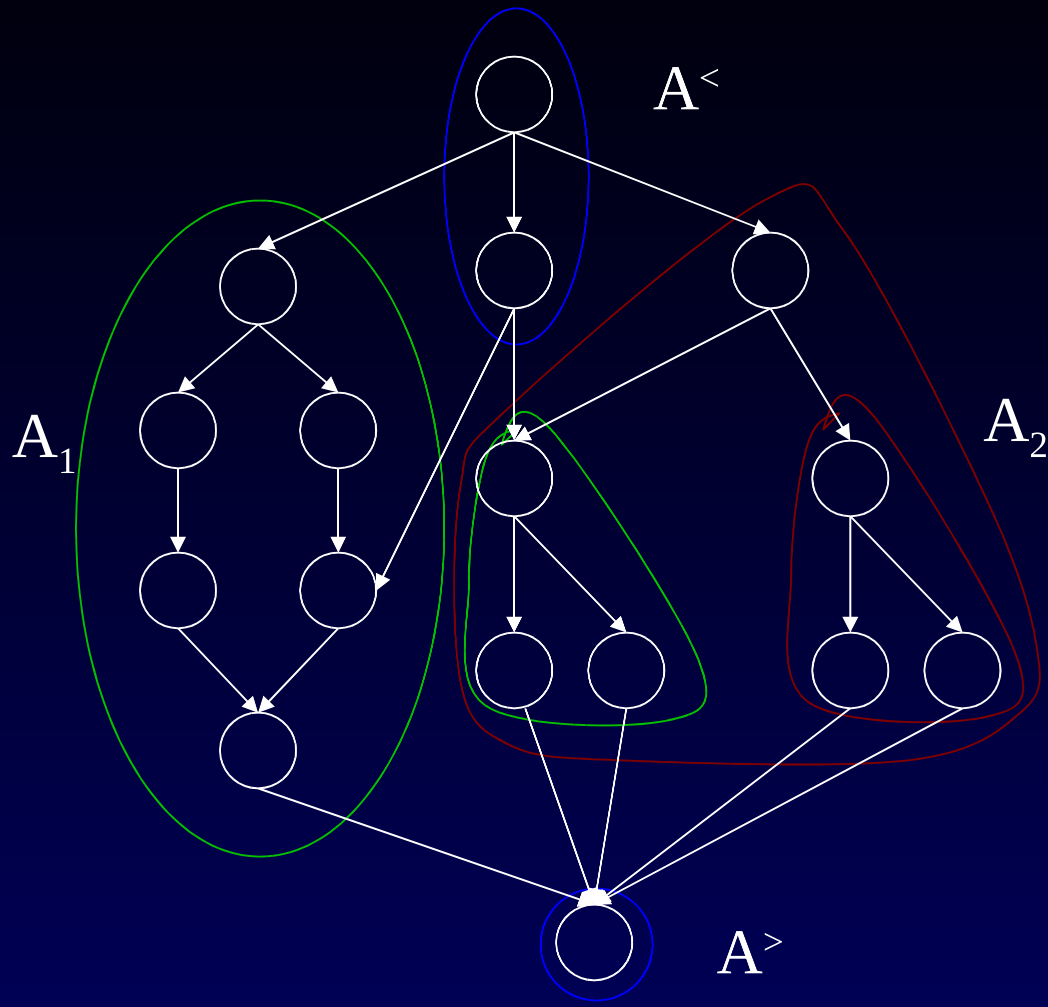
By construction, $\omega_R \leq |A^<| + 2.C + \max(|A_1|,|A_2|) + |A^>|$



as $\min(|A_1|,|A_2|) + \max(|A_1|,|A_2|) = |A_1| + |A_2|$

Thus, $\omega_R \le |V| + 2C - \min(|A_1|,|A_2|) \le 14$

Second level of splitting



« partitioning » DAG problem

- Instance:

  oriented acyclic graph G(V,E)

- Solution:

  two disjoint groups of independent tasks : A_1 et A_2

  such that for all task x in A_1 and y in A_2, there is no path between x and y (and vice versa)

- Objective:

  Maximize the size of the smallest group

  Max $(\min(|A_1|, |A_2|))$

« partitioning » DAG is NP-Complete

- Proof:

From [Garey and Johnson 79]

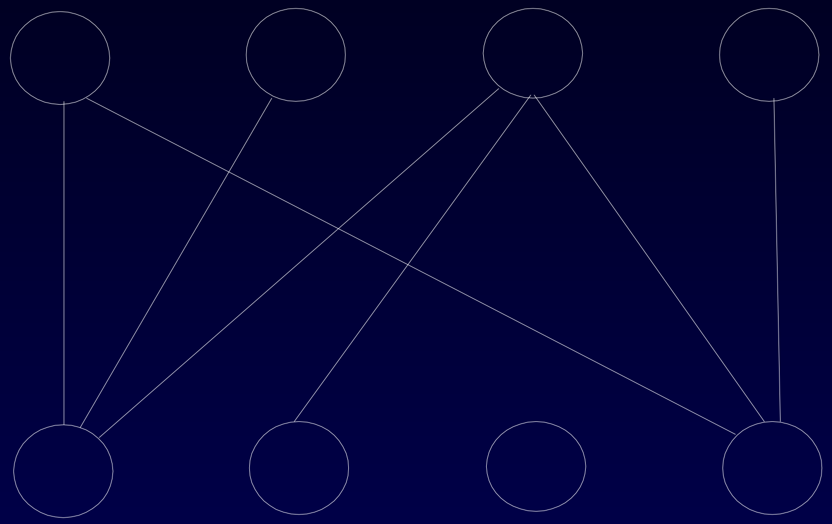
[GT24] BALANCED COMPLETE BIPARTITE SUBGRAPH

INSTANCE: Bipartite graph *G=(V,E)*, positive integer $K \leqslant |V|$

QUESTION: Are there two disjoint subsets $V_1, V_2 \subseteq V$ such that $|V_1| = |V_2| = K$ and such that $u \in V_1, v \in V_2$ implies that $\{u, v\} \in E$?

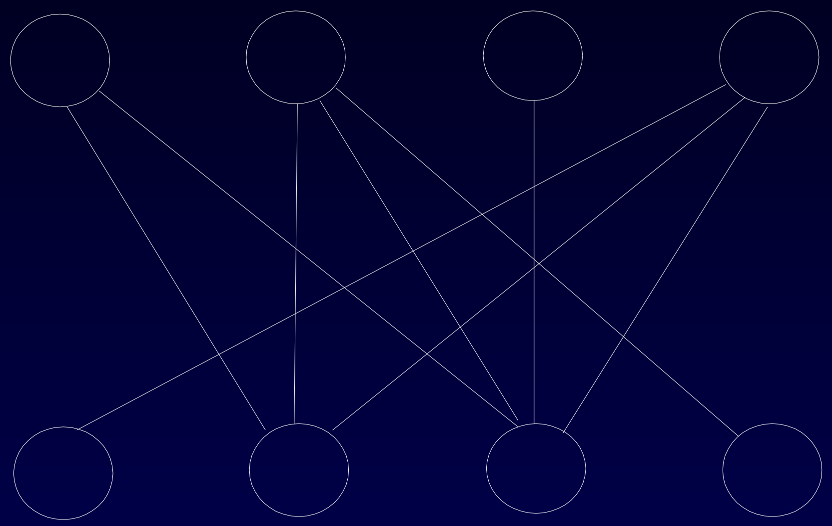
« partitioning » DAG is NP-Complete

Start from a bipartite graph



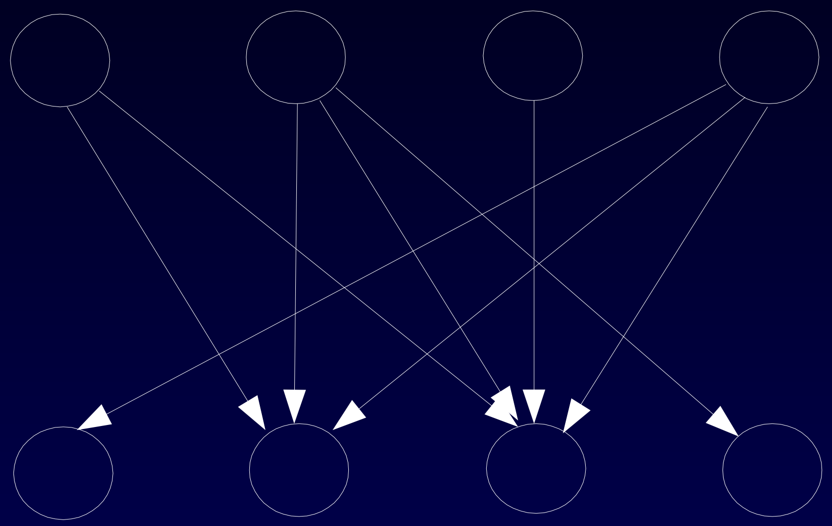
« partitioning » DAG is NP-Complete

Revert all its edges



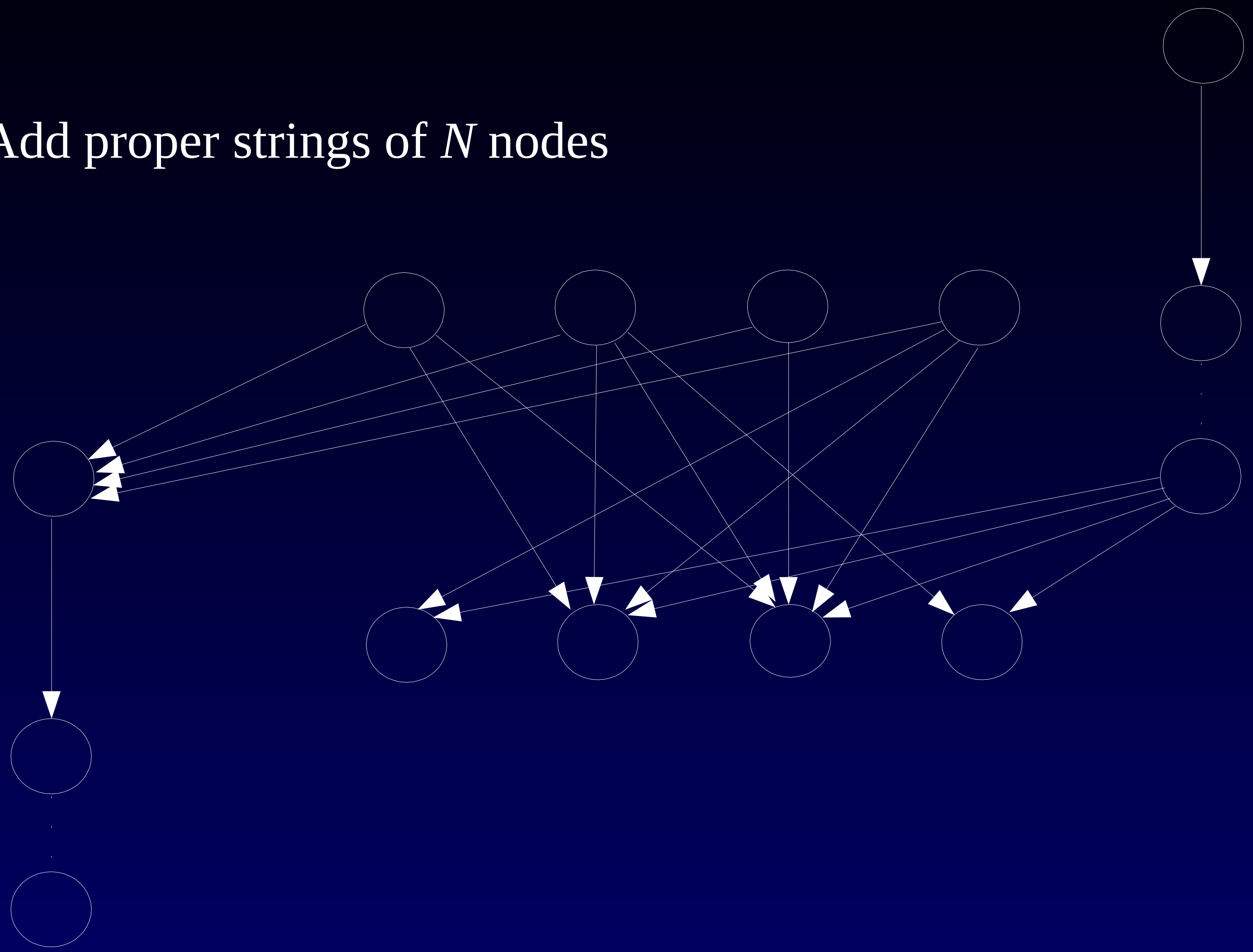
« partitioning » DAG is NP-Complete

Add a direction to them



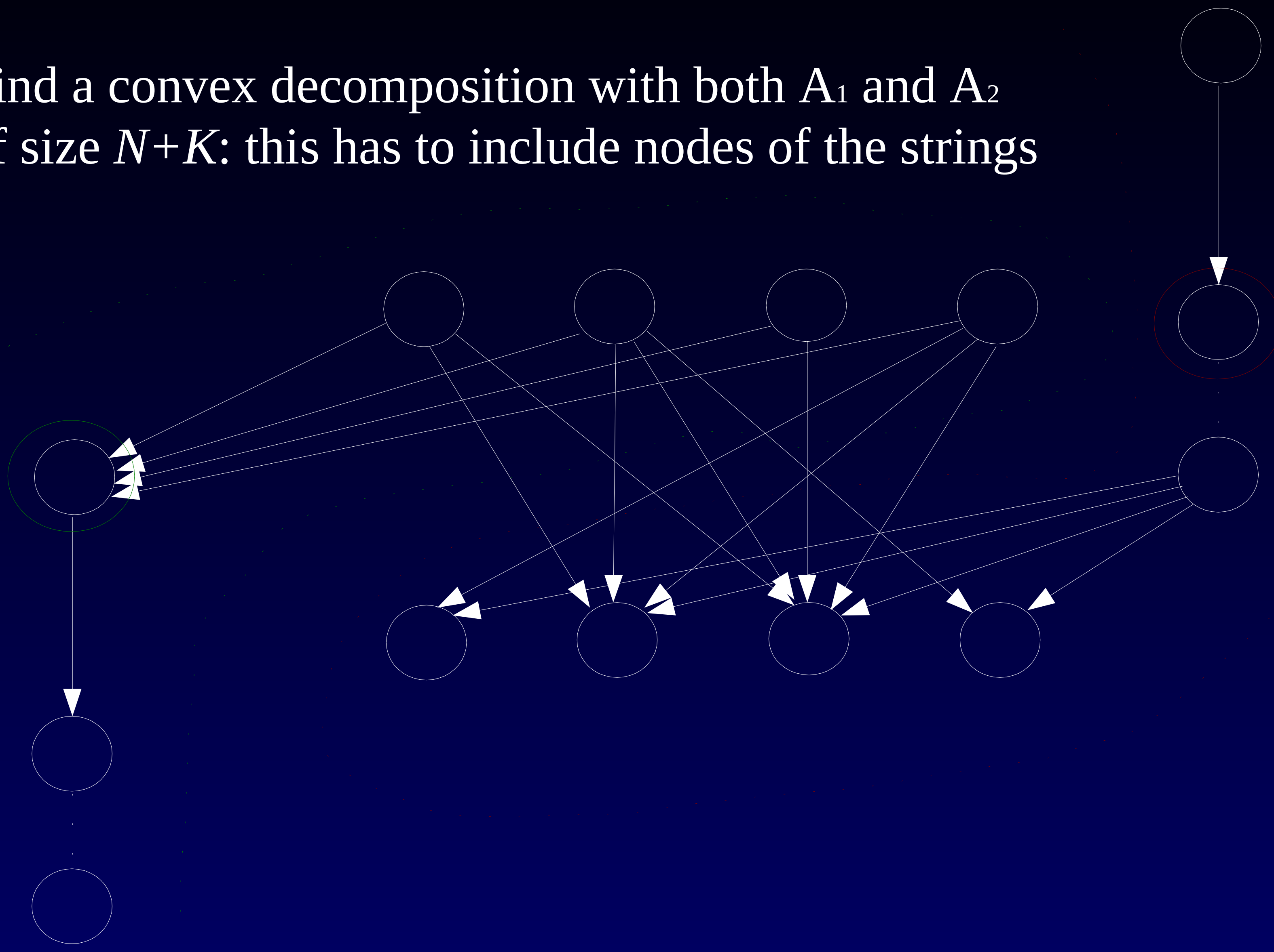
« partitioning » DAG is NP-Complete

Add proper strings of *N* nodes



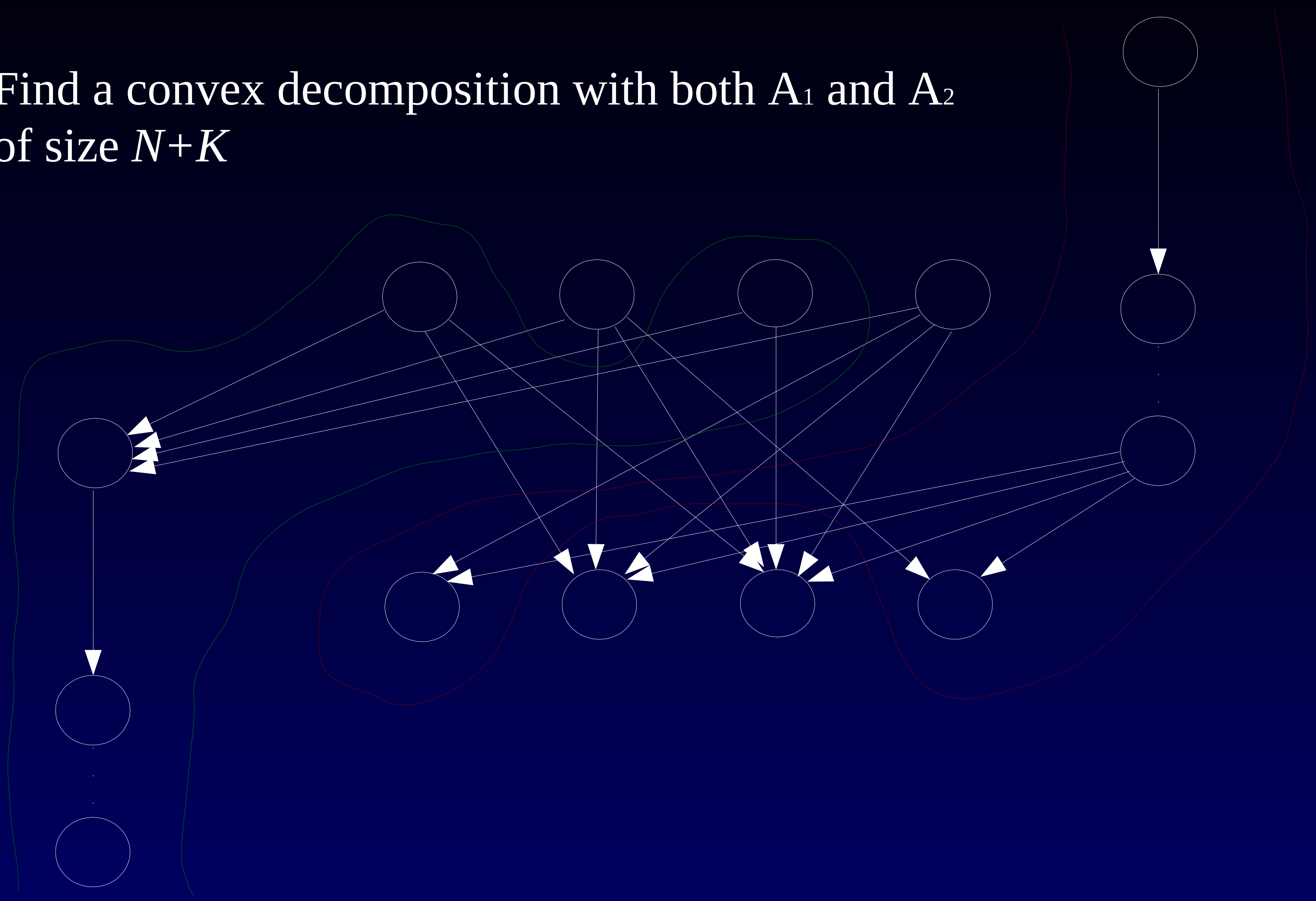
« partitioning » DAG is NP-Complete

Find a convex decomposition with both A_1 and A_2 of size $N+K$: this has to include nodes of the strings



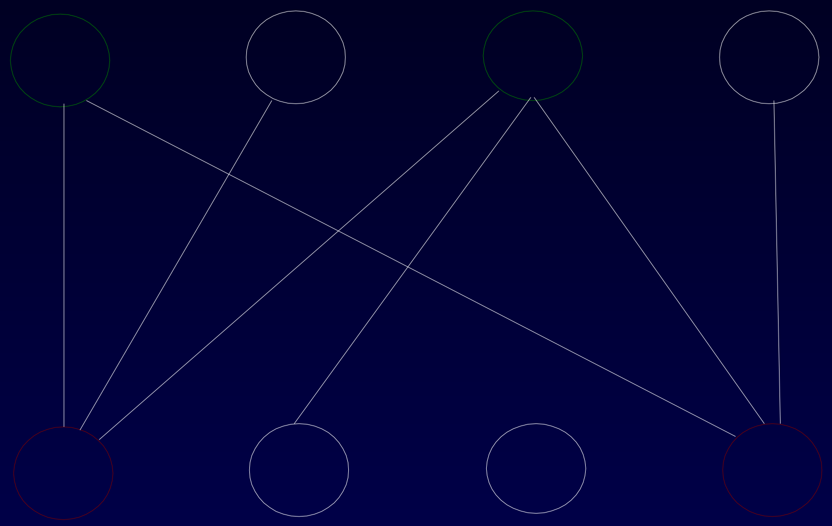
« partitioning » DAG is NP-Complete

Find a convex decomposition with both A_1 and A_2 of size $N+K$



« partitioning » DAG is NP-Complete

This gives us our complete balanced bipartite graph



Linear program

$$
\begin{aligned}
& \max z \\
& \text{such that} \\
& z \leqslant \sum_{u \in V} A_1(u) \\
& z \leqslant \sum_{u \in V} A_2(u) \\
& \forall u \in V,\ A^<(u) + A_1(u) + A_2(u) + A^>(u) = 1 \\
& \forall e = (u, v) \in E,\ A_1(u) + A^<(v) < 2 \\
& \qquad A_2(u) + A^<(v) < 2 \\
& \qquad A^>(u) + A^<(v) < 2 \\
& \qquad A_1(u) + A_2(v) < 2 \\
& \qquad A_2(u) + A_1(v) < 2 \\
& \qquad A^>(u) + A_1(v) < 2 \\
& \qquad A^>(u) + A_2(v) < 2
\end{aligned}
$$

Algorithm for the partitioning tree problem

- By dynamic programming: compute each possible (left,right) couple of possible independant sets sizes
  - Start from leaves, label them whith (1,0)
  - Compute on each node all the possible couples of size repartition
    - use a decreasing order for sizes
    - keep only the dominant couples (greater in all coordinates)
    - at most $O(n^2)$ couples to store on each node
    - proceed child by child (updating the set)
    - three cases to consider for a pair of couples (a,b) and (c,d)

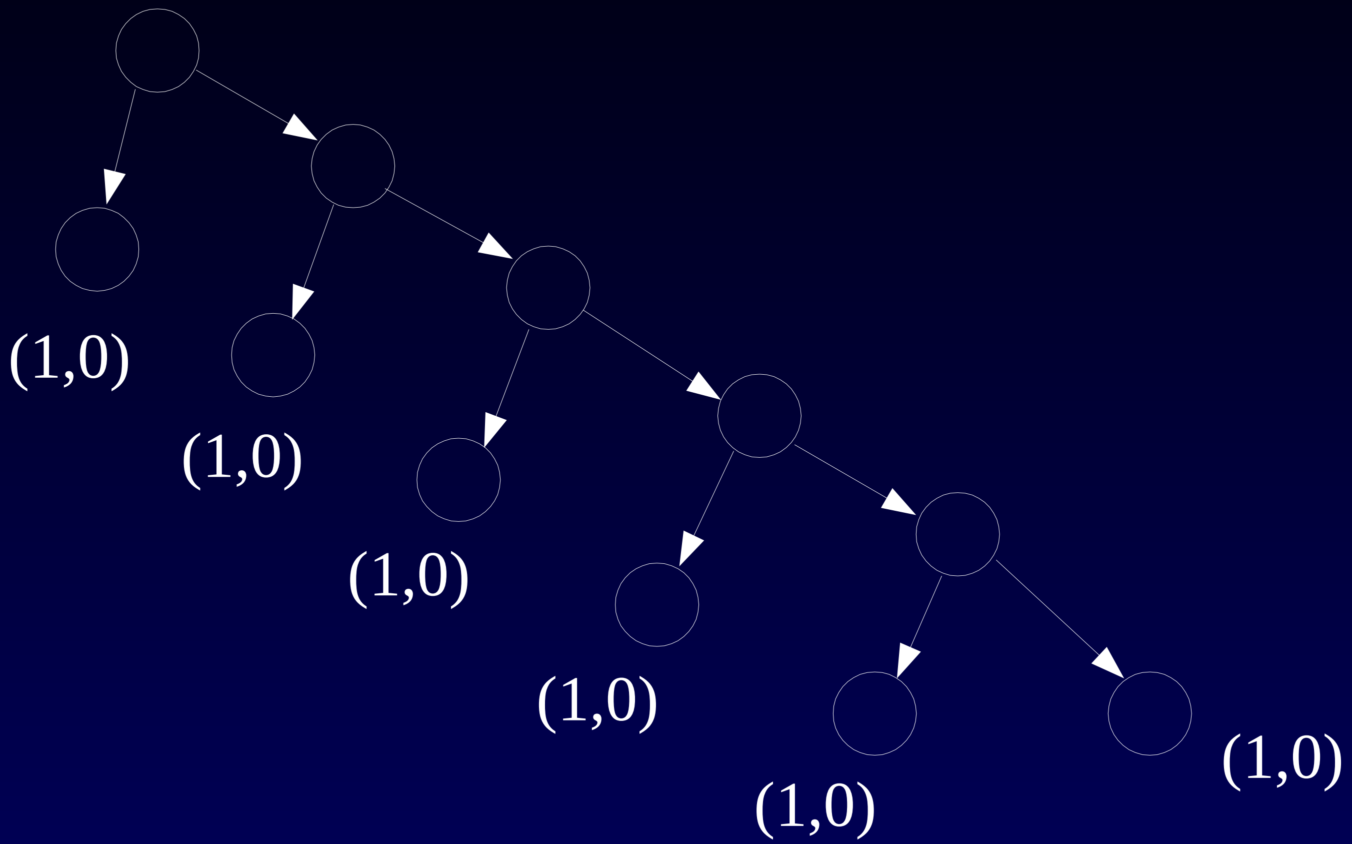
      (a+b+c+d+1, 0) (fuse)

      (a+c, b+d) (combine)

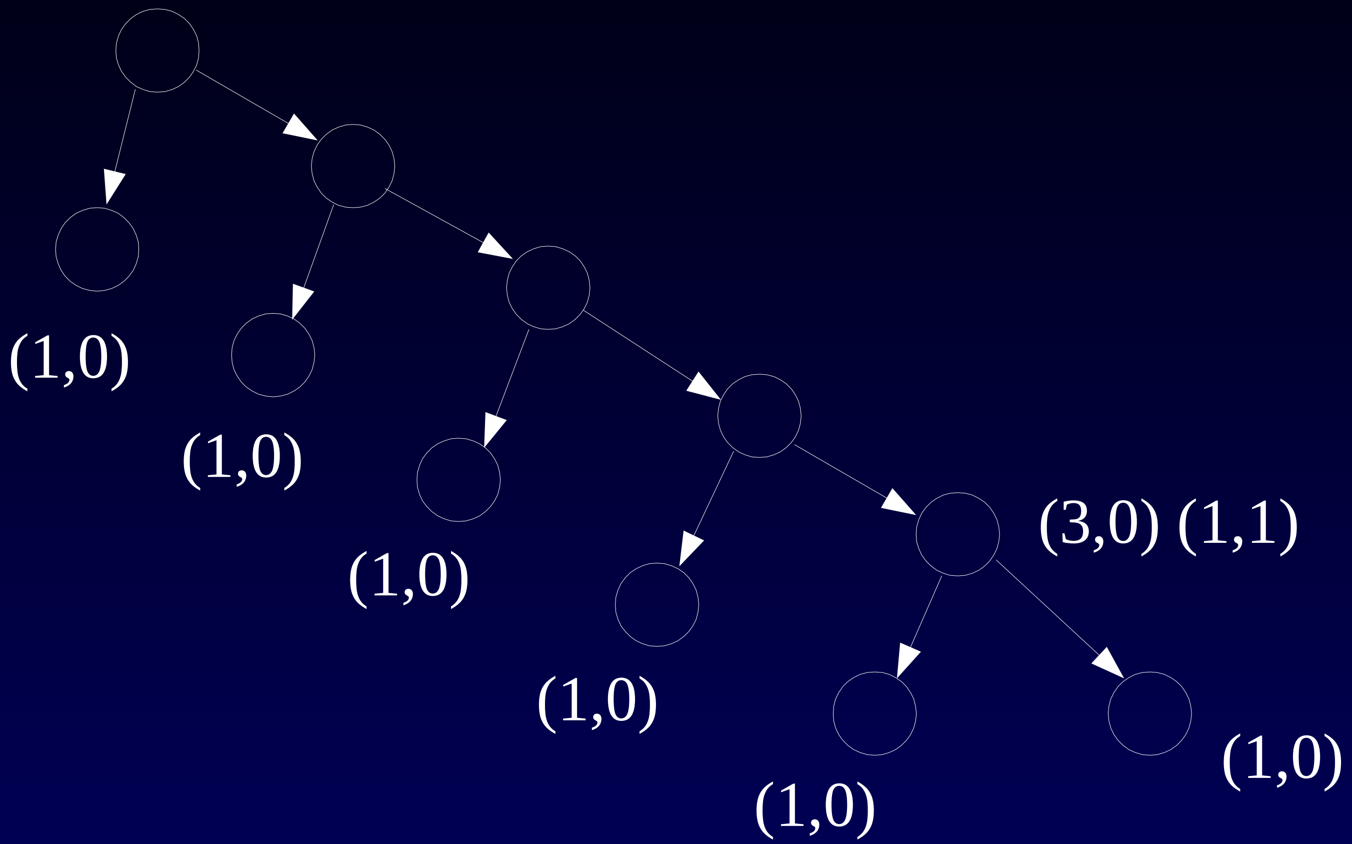
      (a+d, b+c) (cross)

      ... in the proper order

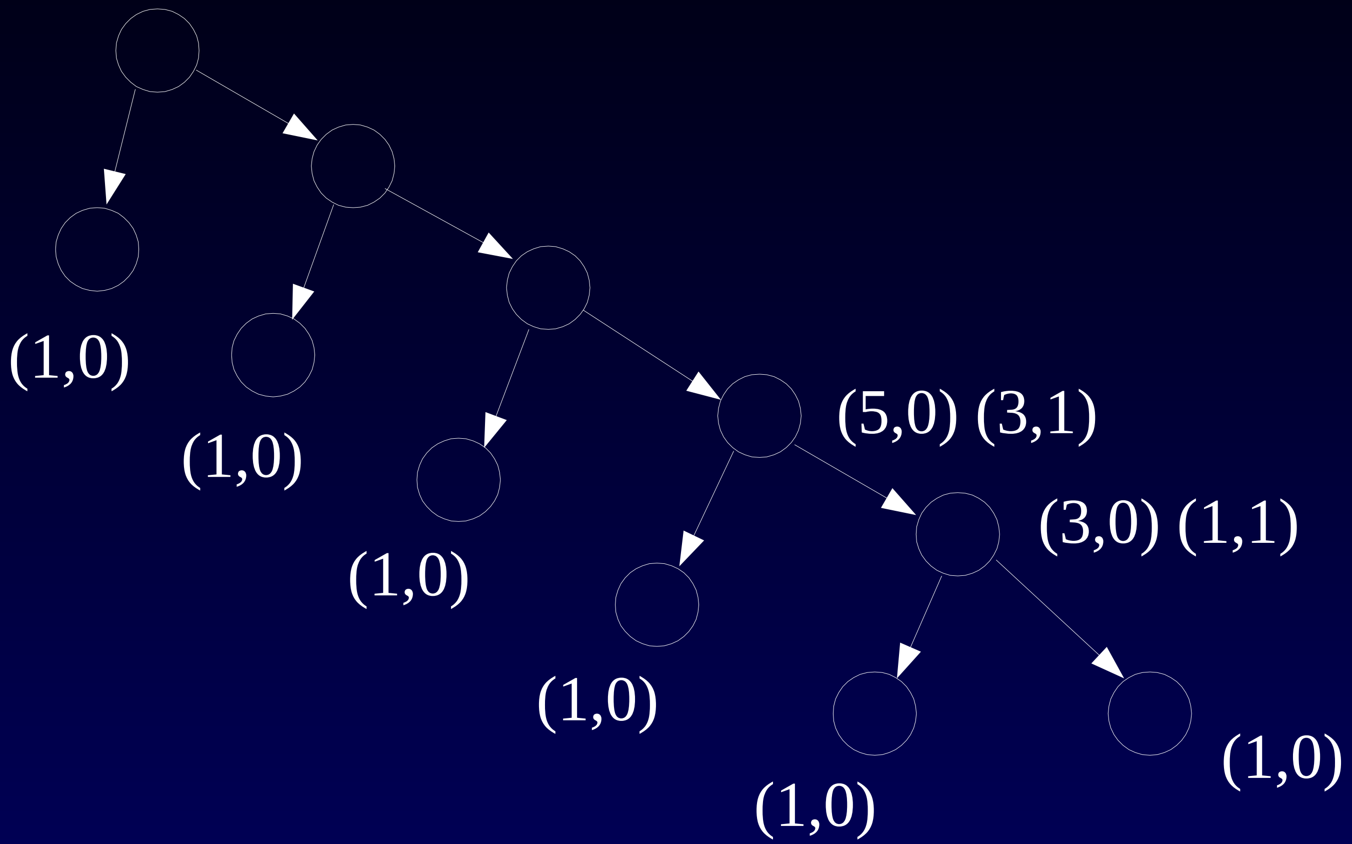
Exemple of partitioning a tree



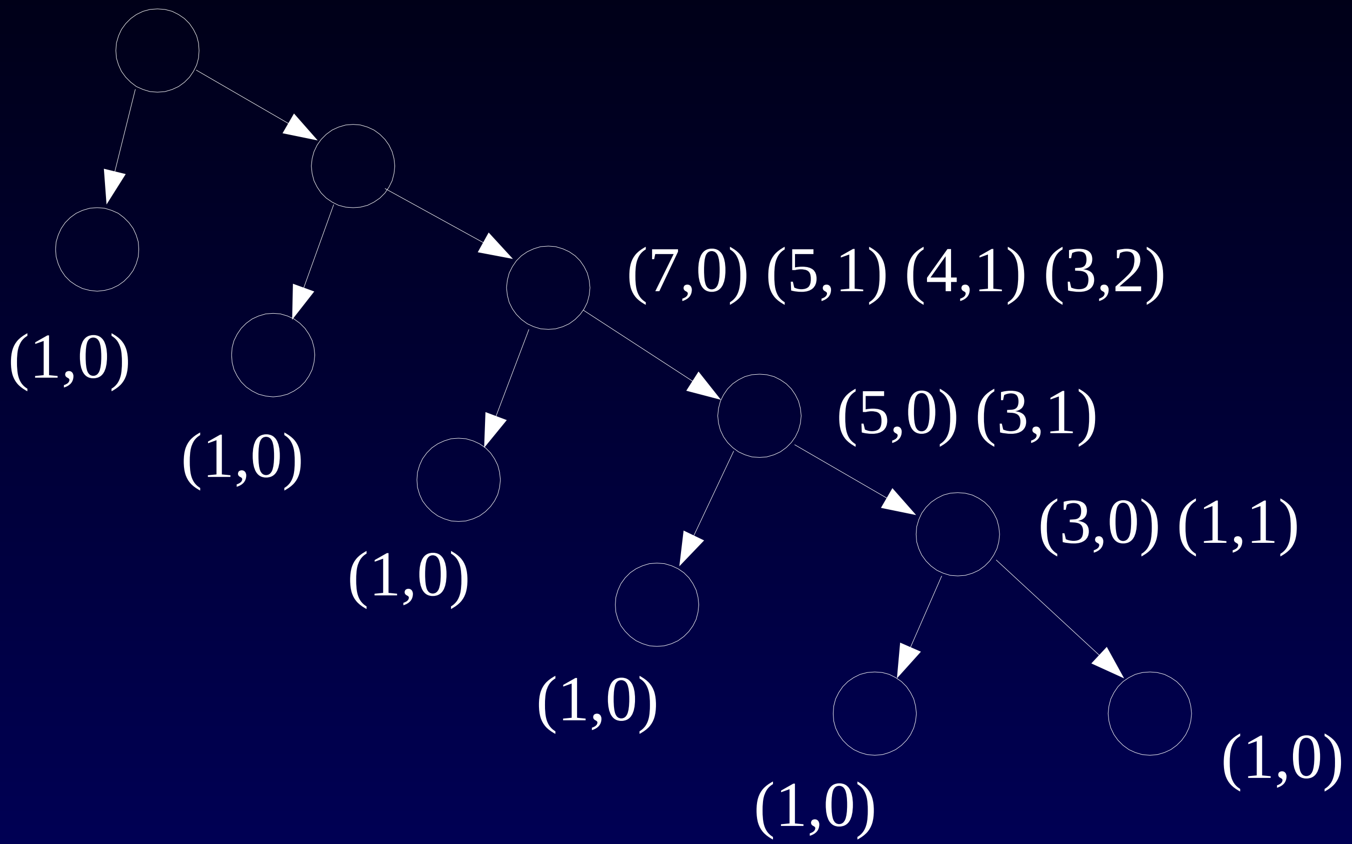
Exemple of partitioning a tree



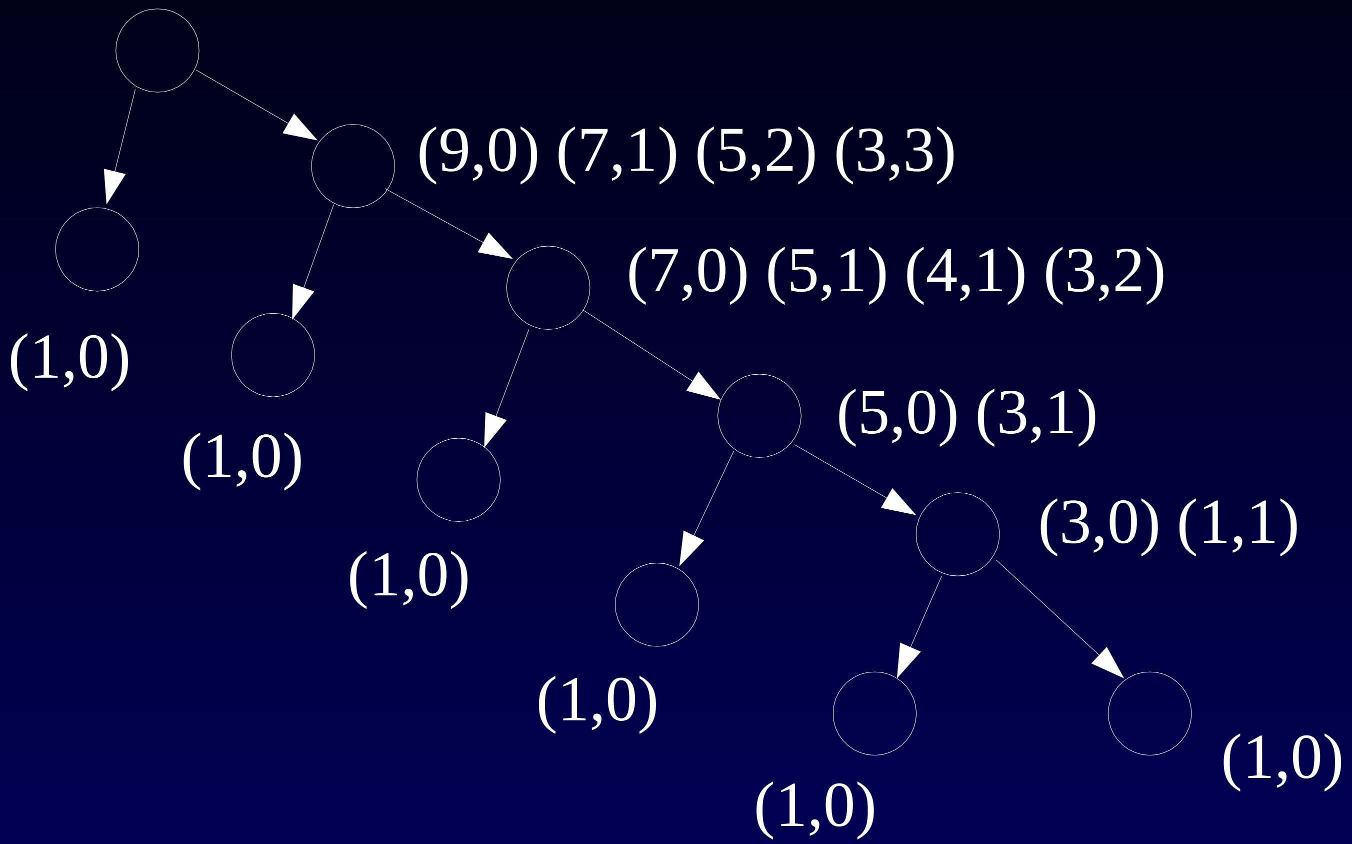
Exemple of partitioning a tree



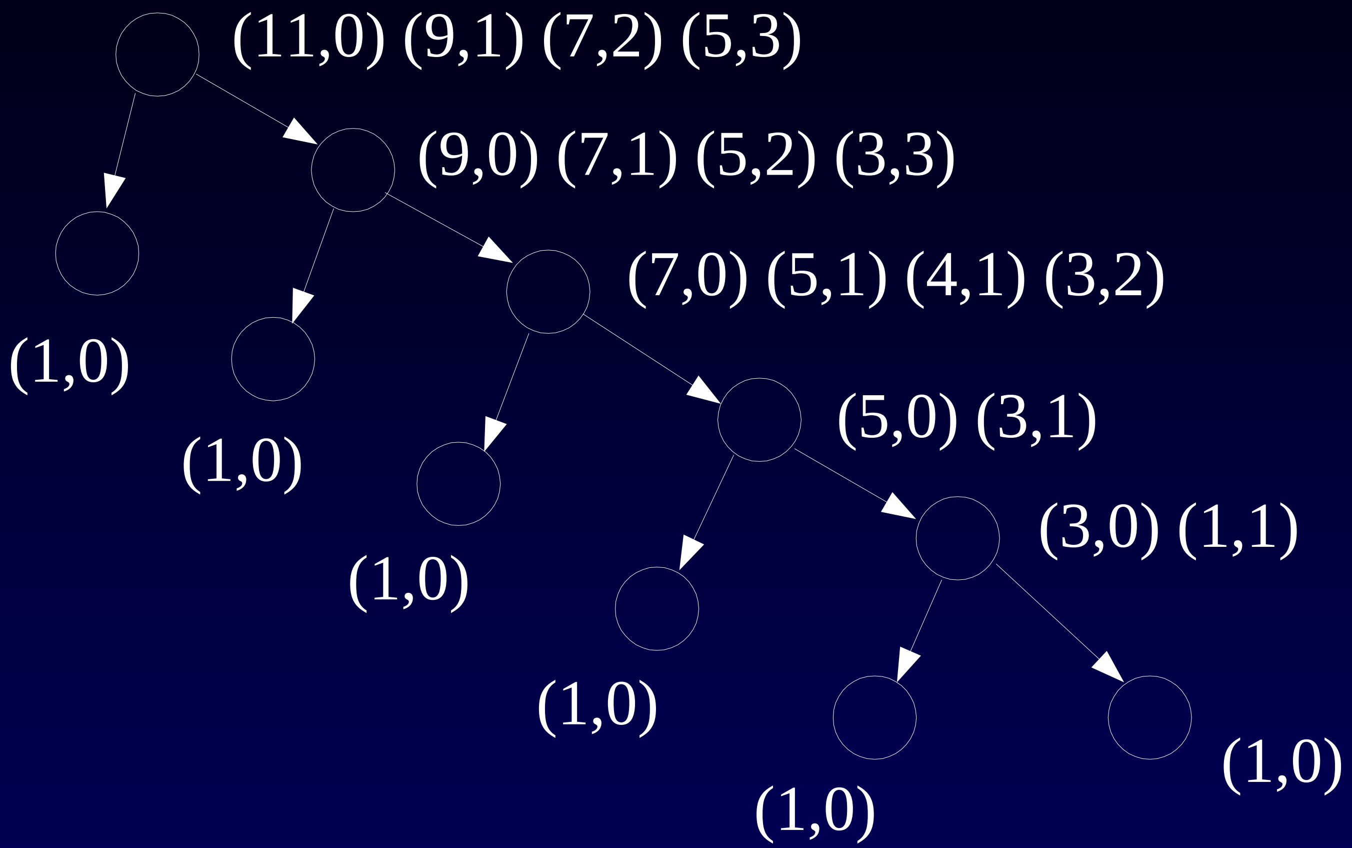
Exemple of partitioning a tree



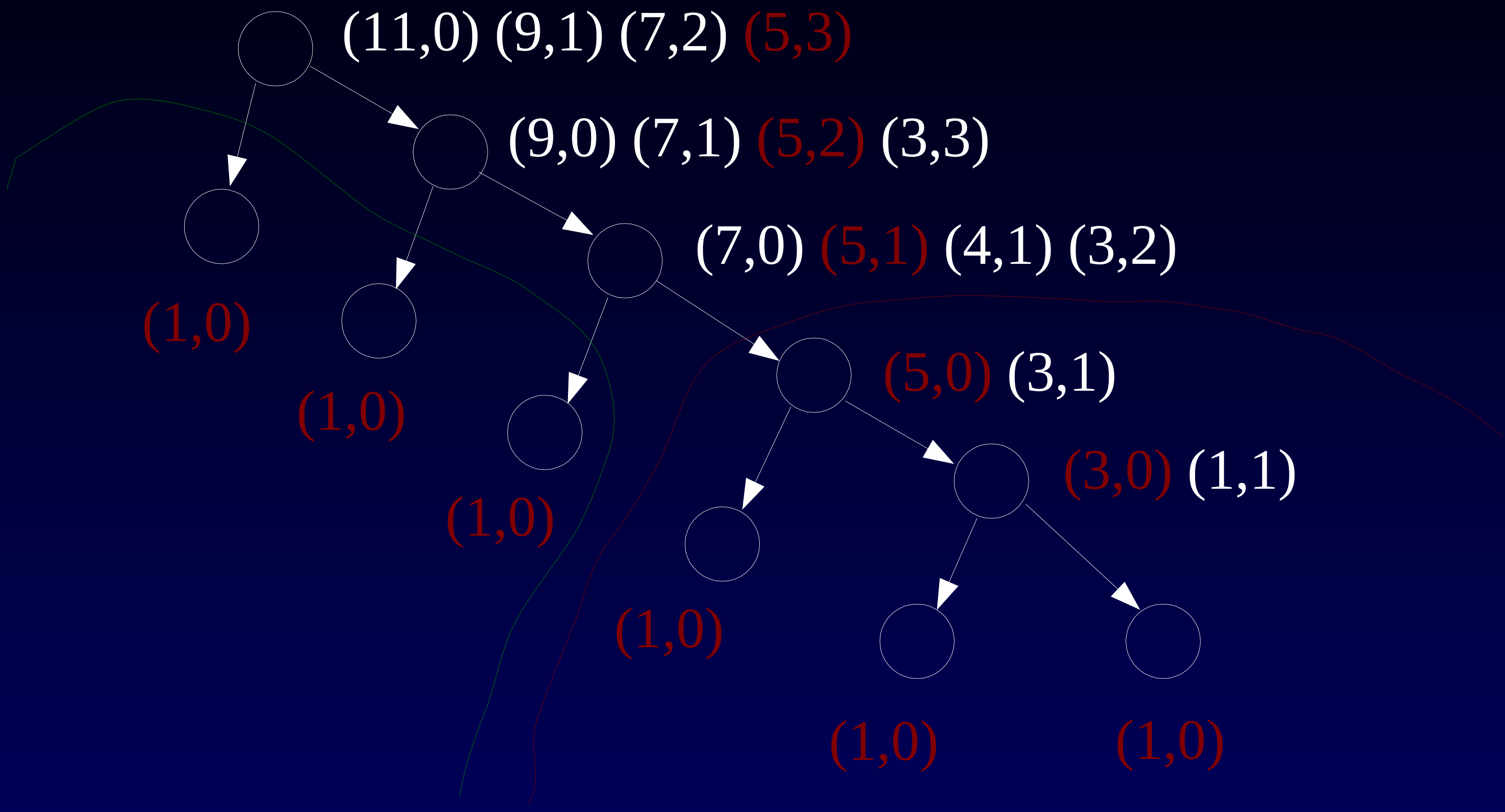
Exemple of partitioning a tree



Exemple of partitioning a tree



Exemple of partitioning a tree



Algorithm for the partitioning DAG problem

Repeat (K times)

Choose a task x

Determine y an independent task from x

Compute both sets Sx = succ(x) and Sy = succ(y)

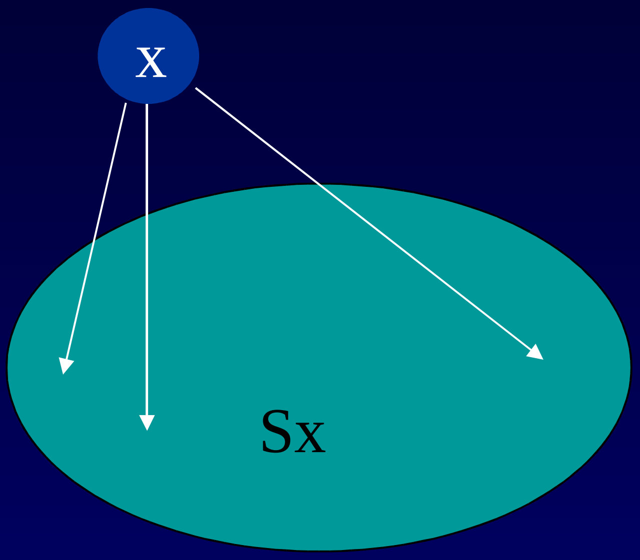
return $A_1 = (Sx \setminus Sy)$ and $A_2 = (Sy \setminus Sx)$

Algorithm from [Trystram and Lepere 2000]

Heuristic for the partitioning DAG problem

Repeat (K times)

- Choose a task x
- Determine y an independent task from x
- Compute both sets Sx = succ(x) and Sy = succ(y)
- return $A_1 = (Sx \setminus Sy)$ and $A_2 = (Sy \setminus Sx)$



Heuristic for the partitioning DAG problem

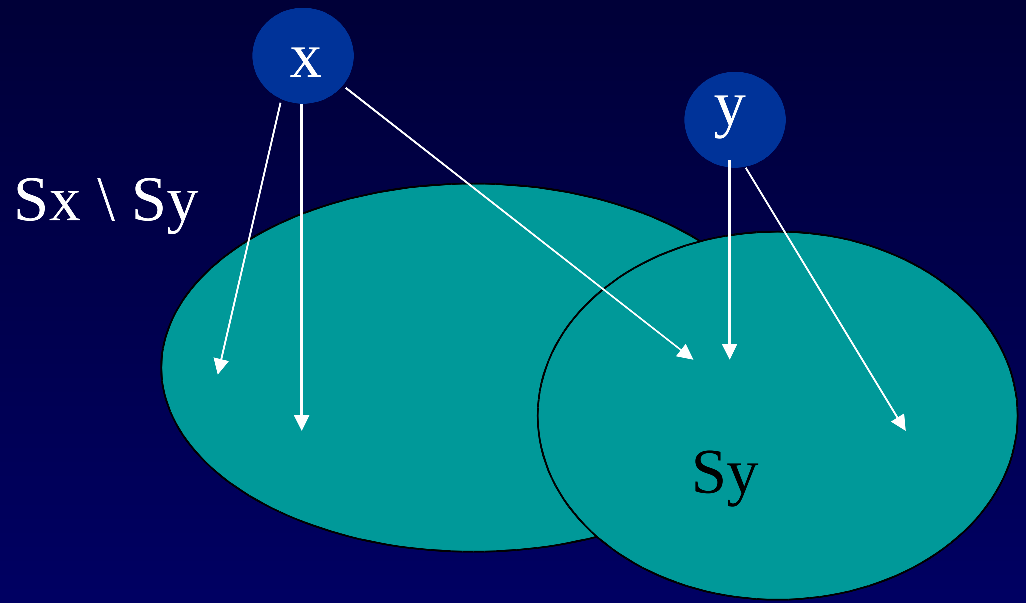
Repeat (K times)

Choose a task x

Determine y an independent task from x

Compute both sets Sx = succ(x) and Sy = succ(y)

return A_1 = (Sx \ Sy) and A_2 = (Sy \ Sx)



Heuristic for the partitioning DAG problem

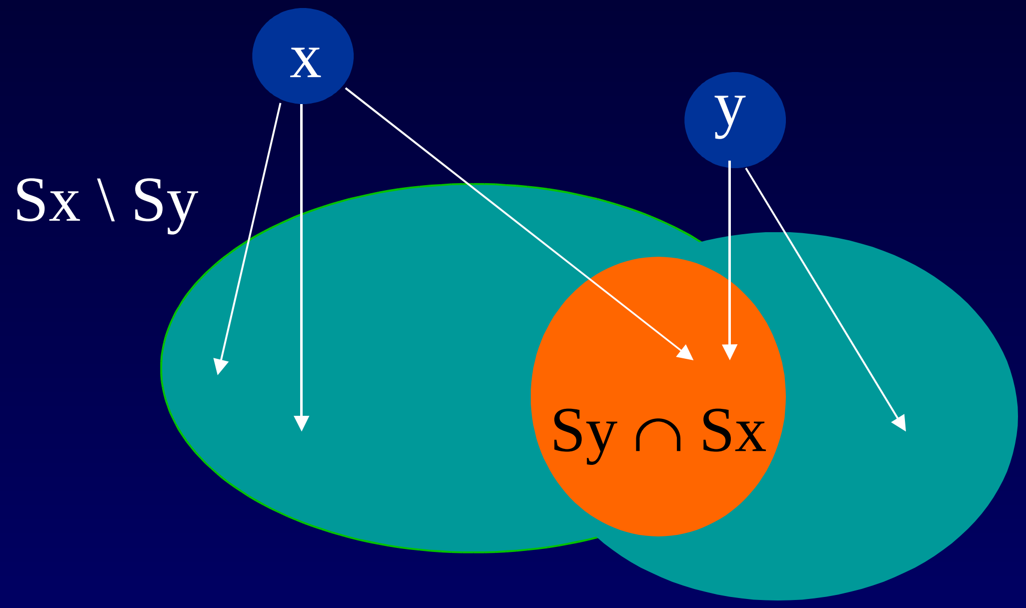
Repeat (K times)

Choose a task x

Determine y an independent task from x

Compute both sets Sx = succ(x) and Sy = succ(y)

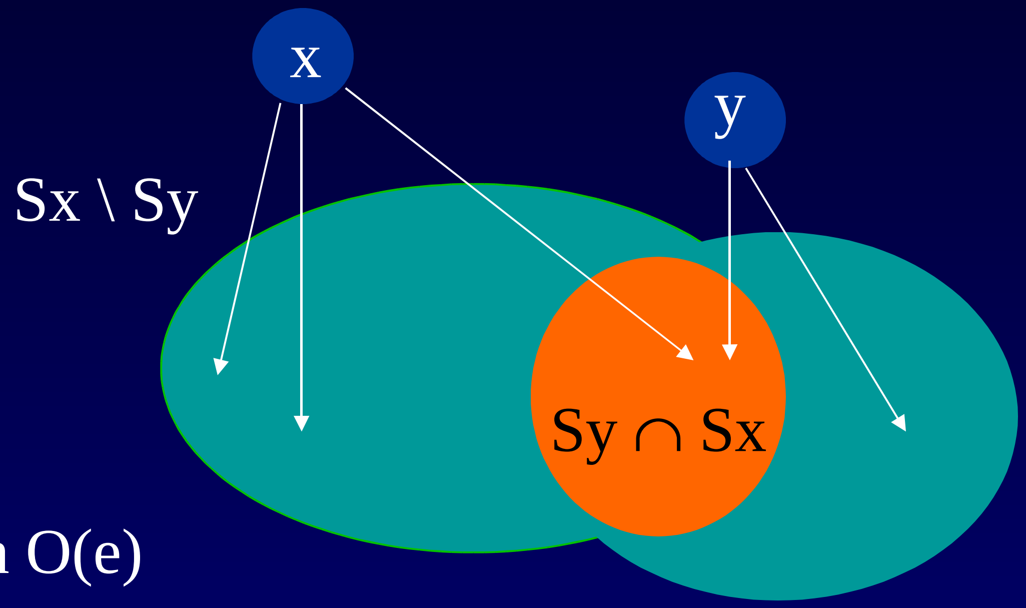
return $A_1 = (Sx \setminus Sy)$ and $A_2 = (Sy \setminus Sx)$



Heuristic for the partitioning DAG problem

Repeat (K times)

- Choose a task x
- Determine y an independent task from x
- Compute both sets Sx = succ(x) and Sy = succ(y)
- return $A_1 = (Sx \setminus Sy)$ and $A_2 = (Sy \setminus Sx)$



Cost of an itération O(e)

Conclusions

- A new approach for clustering:
  - resulting graph is acyclic
  - 2-dominant class of clustering approach
- Some possible recursive decomposition explored:
  - 4 sets division with maximal parallelism is strongly NP-Complete
  - polynomial for trees
  - previous heuristic still valid

Perspectives

- Find a better criterion for recursive decomposition
- Improve the randomized approach with genetic algorithms along with more complex criteria (J. Pecero)
- Combine non unit execution times for tasks with the transition to a coarser grain
- Find a guaranteed algorithm independant from c