Asynchronous Coordination with Constraints and Preferences

Armando Castañeda, Pierre Fraigniaud, Eli Gafni, Sergio Rajsbaum, Matthieu Roy

Asynchronous Coordination with Constraints and Preferences

Armando Castañeda
Instituto de Matemáticas
UNAM
Mexico City, Mexico
castaneda@unam.mx

Pierre Fraigniaud
IRIF, CNRS
University Paris Diderot
Paris, France
fraigniaud@irif.fr

Eli Gafni
CS Dept.
UCLA
Los Angeles, CA, USA
eli@ucla.edu

Sergio Rajsbaum
Instituto de Matemáticas
UNAM
Mexico City, Mexico
rajsbaum@unam.mx

Matthieu Roy
LAAS-CNRS
Université de Toulouse, CNRS
Toulouse, France
roy@laas.fr

ABSTRACT

Adaptive renaming can be viewed as a coordination task involving a set of asynchronous agents, each aiming at grabbing a single resource out of a set of resources totally ordered by their desirability. We consider a generalization of adaptive renaming to take into account scenarios in which resources are not independent.

We model constraints between resources as an undirected graph: nodes represent the ressources, and an edge between two ressources indicates that these two ressources cannot be used simultaneously. In such a setting, the sets of resources that processes may use simultaneously form independent sets. In this note, we focus on this task in a model where such independent sets are computed by wait-free processes.

1. ASYNCHRONOUS COORDINATION AND RENAMING

Adaptive Renaming

In distributed computing, several *tasks* have an *adaptive* version in which the quality of the solution must depend only on the number of processes that participate in a given execution, and not on the total number of processes that may be involved in this task (this number may even be unbounded). A typical example of an adaptive task is *adaptive renaming* [3]: processes acquire distinct *output names* in the space $[1, r]$, where r depends only on the number k of participating processes. In the asynchronous setting with crash-prone processes and read/write registers (the wait-free case), the optimal value for the range is known to be $r = 2k - 1$ [4, 5].

Interestingly, adaptive renaming can also be viewed as a task in which name i is preferred to name j whenever $i < j$. Hence, adaptive renaming can be thought as an abstraction of the problem in which asynchronous *agents* are competing for *resources* totally ordered by their desirability. In other words, adaptive renaming is an abstraction of a problem of *coordination* between agents under *preferences*. Coordination between agents under preferences has been recently investigated in [1, 2] where the *musical chairs* game has been formally defined and solved. In this game, a set of players (modeling the agents) must coordinate so that each player eventually picks one of the available chairs (modeling the resources). Each player initially comes with an a priori *preference* for one chair. In absence of conflict with other players, the player can pick the desired chair, otherwise the conflicting players must coordinate so that they pick different chairs. It has been proven that the smallest number r of chairs for which musical chairs with k players has a solution is $r = 2k - 1$.

Dealing with Constraints on Resources

We foresee that neither adaptive renaming nor musical chairs fully capture typical scenarios of agents competing for resources.

It is often the case that resources are not independent: the literature on scheduling, partitioning or resource allocation — to cite a few— provide several examples in which resources are inter-dependent, causing some resource a not being allowed to be used simultaneously with resource b. That is, using one possibly resource disables others.

In this work, we consider the case in which constraints are modeled as an indirected graph whose nodes are resources, and every edge $\{a, b\}$ indicates that resources a and b cannot be both simultaneously acquired, i.e., acquiring a node disables all its neighbors. In other words, the sets of resources that are allowed are those which form *independent sets* in the constraints graph.

Following this framework, renaming as well as musical chairs tasks correspond to cases where the constraints graph is a stable graph (i.e., a graph with no edges). We address an extension of renaming and musical chairs, targeting an abstraction of a problem of coordination between agents under *preferences* and *constraints*.

PODC'16 July 25-28, 2016, Chicago, IL, USA

ACM ISBN 978-1-4503-3964-3/16/07.
DOI: http://dx.doi.org/10.1145/2933057.2933081

Problem Statement

Definition 1. *In the* coordination with constraints and preferences task CCP*, given an n-node graph $G = (V, E)$ modeling the constraints between the resources in E,*

- *an input is a multiset M of k elements in V, representing the preferences of k processes $p_1, \dots, p_k$,*
- *an output is an* independent set *$I = \{u_1, \dots, u_k\}$ of G, of size k, representing the fact that process p_i acquires u_i, for $i = 1, \dots, k$.*
- *if the input is an independent set of k elements in V, then every process should output its initial preference.*

Seminal papers on renaming [4] and musical chairs [2] showed that in an asynchronous system in which the processes are subject to crash failures, the CCP task is not solvable for k larger than some bound, even for the stable graph G (the value of the bound on k for the stable graph is roughly half the number of nodes of the graph n). Indeed, we are interested in understanding how the addition of constraints may affect the computability of this task.

More precisely, given a graph G, we want to determine the largest k for which the coordination with constraints and preferences task on G is solvable, for any preference multiset M of size at most k. Regarding the computation model, we focus here on *wait-free* systems, in which any subset of processes may fail by crashing, all processes are asynchronous, and communication is performed using read and write operations in shared memory, where each process has its own private registers.

2. LOWER BOUNDS, ALGORITHMS AND OPEN PROBLEMS

A Tight Lower Bound for Paths

Let us first consider the problem for an n-node path P_n. This particular case will enable us to prove a lower bound on the size of Hamiltonian graphs for which the coordination with constraints and preferences task is solvable. Interestingly, this lower bound is almost twice as large as the $2k-1$ bound without constraints resulting from renaming or musical chairs. Specifically, we establish the following:

Theorem 1. *Let k be a positive integer. The smallest integer n for which the coordination with constraints and preferences task in the line of n nodes P_n is solvable for k processes satisfies $n = 4k - 3$. As a consequence, if the coordination with constraints and preferences task in an n-node Hamiltonian graph G is solvable for k processes then $n \geq 4k - 3$.*

The lower bound on n follows from reduction of CCP on a $4k-4$ line to musical chairs with $2k-2$ chairs, i.e., $2k-2$ renaming with initial preference, which is impossible [2]. As sketched in Figure 1, the reduction proceeds by assigning one edge over two to a chair. In the line, each vertex is connected to exactly one bold edge. Bold edges are labelled $1..2k-2$ and correspond to the vertex labelling of the empty graph, and arrows represent the reduction.

The upper bound on n comes from a wait-free algorithm, inspired from the optimal adaptive renaming algorithm in [3], whose main lines are: (1) fix a maximum independent set

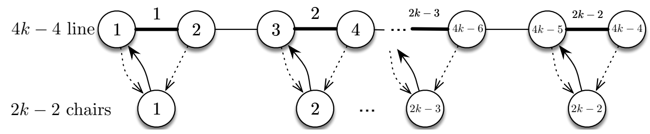

Figure 1: Reduction of CCP task on a $4k-4$ line to renaming on $2k-2$ names.

I in P_n, (2) index the vertices of I from 1 to $2k-1$, and (3) p_i checks if there is no conflict with its initial preference; if p_i is not in conflict, the it decides its initial preference, otherwise it runs an optimal (adaptive) renaming algorithm on these indexes. This algorithm is *static* in the sense that I is predetermined in advance and not during an execution. Algorithm 1 describes such a generic static algorithm.

From this preliminary result on P_n, one may think that solving the coordination with constraints and preferences task in a graph G boils down to classical renaming once a maximum independent set in G is fixed. We show that this is *not* the case. In fact, even for an instance as simple as the n-node ring C_n, the problem becomes highly non trivial.

Rings: the Case of the Pentagon

Theorem 2. *Let k be a positive integer. The smallest n for which the coordination with constraints and preferences task in C_n is solvable for k processes satisfies $4k - 3 \leq n \leq 4k - 2$.*

The lower bound is a consequence of Theorem 1 since C_n is Hamiltonian. A quite intriguing fact is that the wait-free algorithm derived from an adaptation of an optimal algorithm for classical renaming run on a maximum independent set of C_n does not match the lower bound, and is off by an additive factor $+1$.

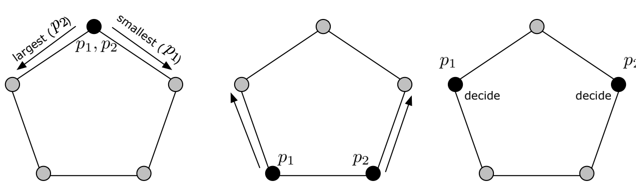

Figure 2: Optimal algorithm for two processes in the pentagon: rules when two processes are executing

The algorithm for two processes in the pentagon C_5 is depicted on Figure 2, which represents the snapshot of a process, and the action to take (represented as arrows) based on this snapshot when 2 processes participate. There are three cases, depending on whether the two processes are currently occupying nodes at distance 0, 1, or 2. Notice that if the snapshot reveals that the process is alone, then it decides the node that it currently occupies, i.e., its preferred node. If the snapshot reveals that the two processes occupy the same node, then the action depends on the ID: going clockwise for the process with smallest ID, and counterclockwise otherwise. If the snapshot reveals that the two processes occupy two neighboring nodes, then the action is: going away from the other node. Finally, if the snapshot reveals that the two processes occupy two nodes at distance 2, then the action is to decide the currently occupied node.

We believe that the difference of 1 between the lower and upper bounds for C_n is certainly not anecdotal, but is the witness of a profound phenomenon that is not yet understood, with potential impact on classical renaming and musical chairs. The main outcome of this paper is probably the observation that *static* algorithms, i.e., algorithms based on *fixed* precomputed positions in the graph of constraints, might be sub-optimal by allocating less resources than the optimal. The optimal algorithm for coordinating two processes in the pentagon is not static, and the set of allocated resources output by this algorithm spans *all possible* independent sets. The design of optimal *dynamic* (i.e., non static) algorithms for solving the coordination with constraints and preferences task appears to be a challenge, even in the specific case of the cycle C_n.

The Inherent Difficulty of the General Case

The inherent difficulty for asynchronous crash-prone processes to coordinate under constraints and preferences, even in graphs with arbitrarily large independent sets, can also be illustrated by the complete bipartite graph $K_{x,y}$ with $n = x + y$ nodes. We show that, although $K_{x,y}$ has very large independent sets (of size at least $\min\{x, y\}$), processes cannot coordinate *at all* in this graph.

THEOREM 3. *Let x, y be positive integers. Coordination with constraints and preferences in the complete bipartite graph $K_{x,y}$ is unsolvable for more than one process.*

A Suboptimal Static Algorithm for General Graphs

Finally, on the positive side, given any graph G, we can design a static algorithm ALG solving the coordination with constraints and preferences task in G —recall that static, in this case, means that ALG is based on a statically defined independent set I, a priori known to processes.

More precisely, ALG requires a *k-admissible* independent set: given $G = (V, E)$, an independent set I of G is k-admissible if for every $W \subseteq V$ of size at most $k - 1$, we have $|I \setminus N[W]| \geq |I \cap W| + 1$ where $N[W]$ denotes the set of nodes at distance at most 1 from a node in W ($N[w] = \{w\} \cup \{v \in V : \{v, w\} \in E\}$, and $N[W] = \cup_{w \in W} N[w]$). Figure 3 provides a sketch of k-admissibility.

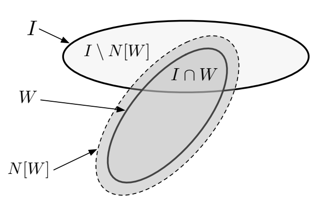

Figure 3: k-admissibility : $|I \setminus N[W]| \geq |I \cap W| + 1$, if $|W| \leq k - 1$

We can prove that among static algorithms, ALG is optimal, which completely closes the problem for static algorithms.

THEOREM 4. *Let G be a graph, and k be a positive integer. Let I be a k-admissible independent set in G. Then,* ALG *instantiated with I solves the coordination with constraints and preferences task in G with k processes. Moreover, if G has no $(k+1)$-admissible independent set, then no static algorithm can solve the coordination with constraints and preferences task in G with more than k processes.*

Algorithm 1 $G = (V, E)$ is a graph, and I is an ordered independent set in G. Code for p_i.

function *CoordinationConstraints*($u_i \in V$: initial preference)
$cur_i \leftarrow u_i$
loop
  write(cur_i)
  snapshot memory to get $view = \{cur_{j_1}, \ldots, cur_{j_r}\}$
  $view' \leftarrow view \setminus \{cur_i\}$ ▷ remove cur_i from *view*
  if $view' \cap N[cur_i] = \emptyset$ **then** ▷ check for conflicts
    return cur_i ▷ no conflict ⇒ decide cur_i
  else ▷ conflict ⇒ compute a new position
    $free \leftarrow I \setminus N[view']$ ▷ rule out conflicts from I
    $\ell \leftarrow |\{s : cur_{j_s} \in I \text{ and } j_s < i\}| + 1$ ▷ ranking
    $cur_i \leftarrow \ell^{th}$ element in *free* ▷ next proposal
  end if
end loop

Algorithm 1 is the pseudocode of ALG. The algorithm uses a shared array *view*, accessed with write and snapshot operations, where each entry is initially $\perp$. The local variable cur_i stores the current proposal of process p_i. Algorithm 1 is a rewriting of a textbook renaming algorithm for shared memory [4], adapted to take into account the statically fixed independent set I and the constraint that no two process may end up in connected vertices.

Conclusion

In this note, we introduced the coordination with constraints and preferences task, a task that models a set of processes competing for interdependent resources. We sketched a lower bound for Hamiltonian graphs, and provided an optimal algorithm for 2 processes on a pentagon. If processes agree beforehand on a given maximal independent set, we describe a *static* algorithm for solving this problem. Static algorithms are in general sub-optimal, as illustrated on the pentagon.

Hence, the design of optimal *dynamic* algorithms for solving the coordination with preferences and constraints tasks in graphs appears to be a challenging open problem, even for the relatively simple case of rings.

3. REFERENCES

[1] Y. Afek, Y. Babichenko, U. Feige, E. Gafni, N. Linial, and B. Sudakov. Oblivious collaboration. In D. Peleg, editor, *Distributed Computing*, volume 6950 of *Lecture Notes in Computer Science*, pages 489–504. Springer Berlin Heidelberg, 2011.

[2] Y. Afek, Y. Babichenko, U. Feige, E. Gafni, N. Linial, and B. Sudakov. Musical chairs. *SIAM Journal on Discrete Mathematics*, 28(3):1578–1600, 2014.

[3] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524–548, 1990.

[4] H. Attiya and J. Welch. *Distributed Computing Fundamentals, Simulations, and Advanced Topics, Second Edition.* John Wiley and Sons, Inc., 2004.

[5] E. Gafni, A. Mostéfaoui, M. Raynal, and C. Travers. From adaptive renaming to set agreement. *Theoretical Computer Science*, 410(14):1328 – 1335, 2009. Structural Information and Communication Complexity (SIROCCO 2007).