

BREAKING SYMMETRIES IN DISTRIBUTED CONSTRAINT PROGRAMMING PROBLEMS

Xavier OLIVE

Hiroshi NAKASHIMA

Graduate School of Informatics, Kyoto University

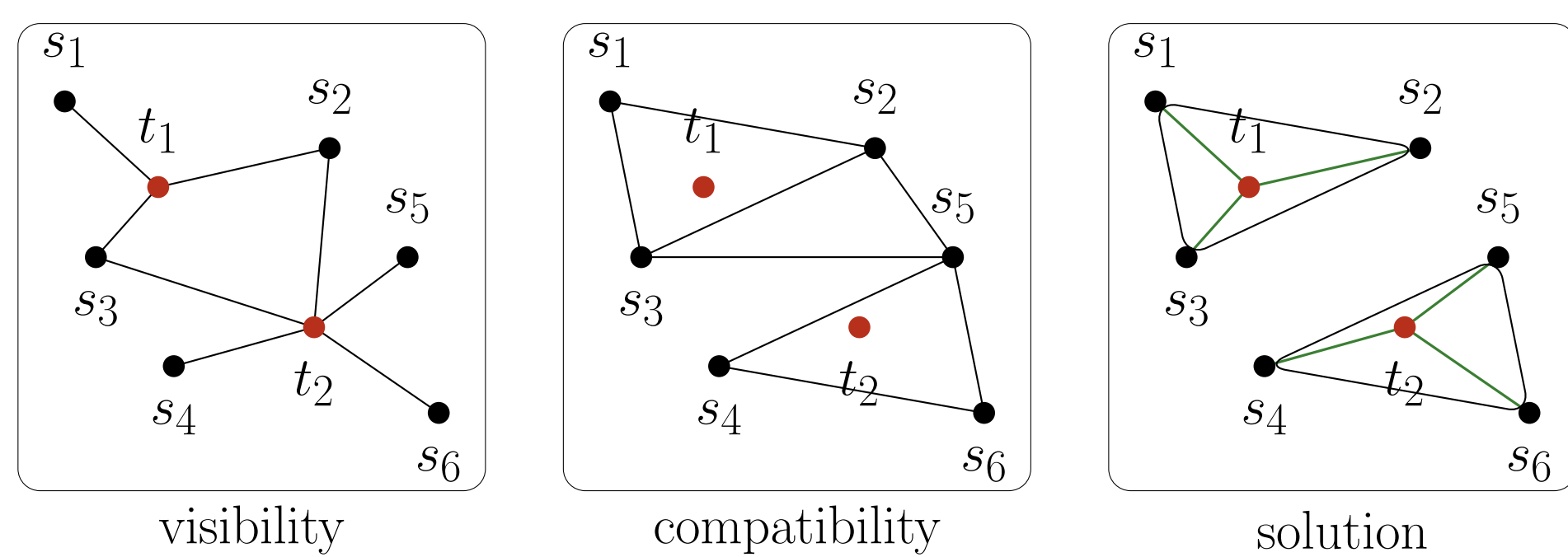
Abstract

Though various preprocessing techniques have been studied for improving the performance of distributed constraint satisfaction problems, no preprocessing technique for detecting and breaking symmetries has been studied in depth.

In this poster, we describe a method for detecting some symmetries of a given distributed problem and for exploiting them. Then, we validate this method as a preprocessing method for ADOPT and DPOP algorithms for some instances of the SensorDCSP problem, to find our symmetry breaker improve their performance up to 1.7 and 1.8 times respectively.

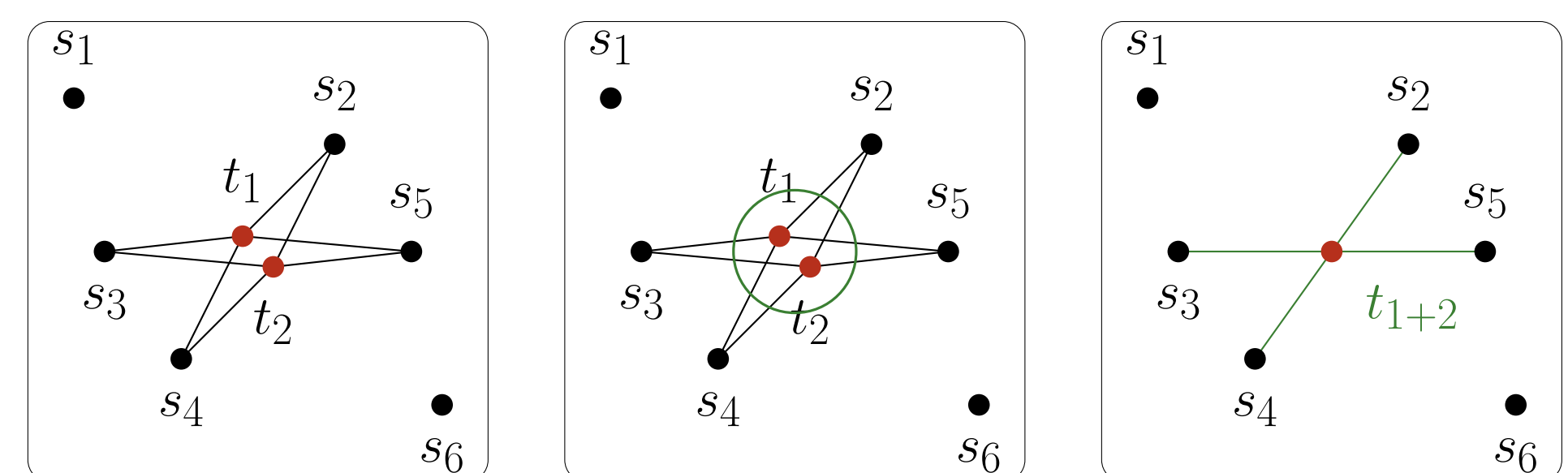
DCSP

- A DCSP is defined as a finite set of variables x_1, \dots, x_n , a set of domains d_1, \dots, d_n , a list of agents a_1, \dots, a_n , and a set of constraints c_1, \dots, c_k . Each constraint has a scope of variables, thus a scope of agents.
- Constraints can be local (private) or global (distributed).
- A solution to a DCSP is a global assignment violating none of the constraints.
- **SensorDCSP** is a benchmark for DCSP problems: given multiple sensors s_i and multiple mobiles t_i to be tracked by the sensors, the goal is to allocate three sensors to track each mobile node, subject to visibility and compatibility constraints.



Symmetries

- A symmetry is “a mapping that permute the variables of a problem by pair, while leaving the constraint unchanged”. Here, if two mobiles are very close to each other, the same constraints apply to t_1 and t_2 , so we can redefine the problem with a virtual mobile t_{1+2} .

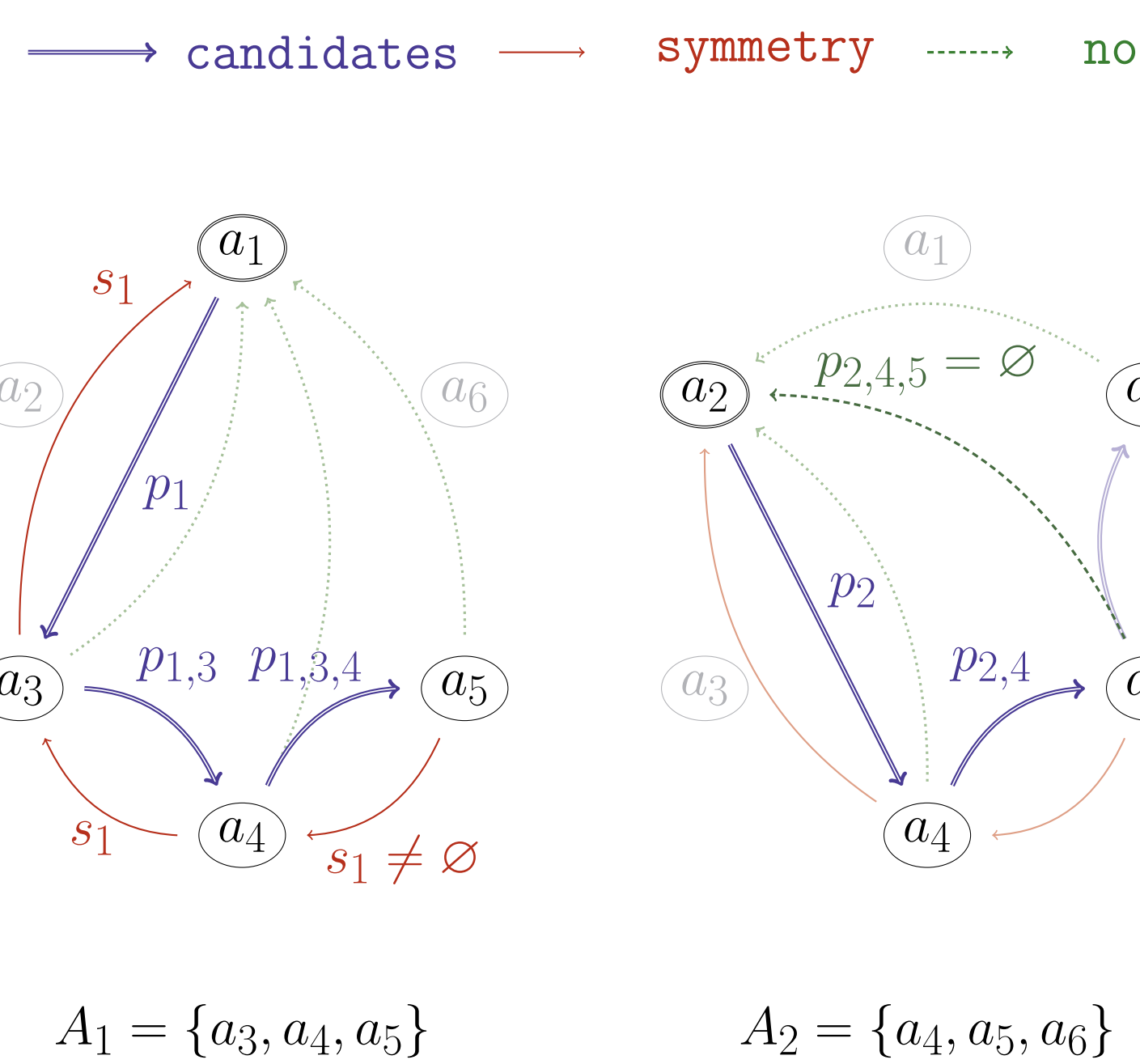


- Each sensor/agent can detect which mobiles are within range. For s_2 , t_1 and t_2 are interchangeable, so $t_1 \leftrightarrow t_2$ is a local symmetry. But s_2 alone does not know whether t_1 and t_2 are interchangeable for the other sensors (global symmetry). More generally, no agent can know the entire problem and detect the global symmetries without communication with the other agents about local symmetries.

Symmetry Detection

Algorithm

- We set a priority order on the agents.
- Each agent a_i detects its local symmetries, and keeps in p_i only the ones who involve variables owned by itself, or agents of lower priority (named A_i).
- Each agent a_i initiates a unique process and sends both A_i and p_i to the first agent in A_i .
- When an agent a_j receives p_i , it builds $p_{i,j}$ by taking out of p_i the symmetries which are not leaving the set of local constraints of a_j invariants. Then,
 - * if $p_{i,j} = \emptyset$, we stop the process and send a **no** to the agent a_i who initiated the process;
 - * if a_j is the last agent in A_i , we send $p_{i,j}$ back the message the way it came;
 - * otherwise, we send $p_{i,j}$ to the next agent in A_i .



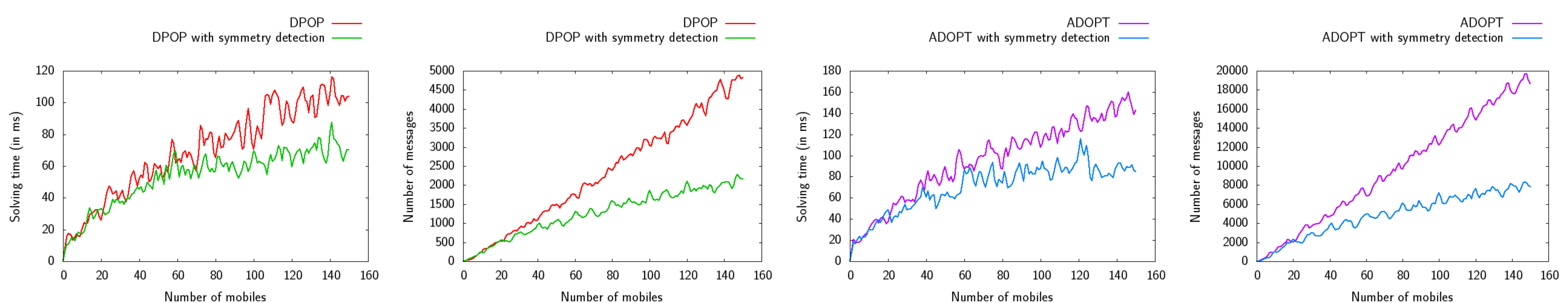
- With **SensorDCSP**, each sensor sends the mobiles it detects to the sensors it shares a constraint with, and they check whether they detect the same mobiles.

- **Termination** Each agent starts one process, ending with the reception of either a **no** or a **symmetry** message. At the reception of this last message, it sends a **done** message to a controller who will notice all the agents that they can start processing the symmetries, e.g. reformulating the problem.

- **Complexity** The preprocessing initiates a number of messages in $O(n \cdot k)$ for n agents having $k = \max |A_i|$ neighbours at most. This number should be insignificant compared to the communication we save thanks to symmetry breaking.

Results and Perspectives

- Preprocessing algorithm executed on a 25-sensor n -mobile **SensorDCSP** problem, and compared the performance with regular DPOP and ADOPT. (Core2Duo Linux PC, 2GB Ram)



- For the biggest problem, DPOP (resp. ADOPT) sees its execution time improved by about 30% (resp. 40%). As for the number of sent messages, it is cut down by some 50%.

Perspectives

- Could we ensure that the effort spent for detecting the symmetries is really worth the try, considering what we save after redefining the problem with symmetries?
- Could we anticipate the effectiveness of symmetry breaking instead of potentially wasting time?
- More experiments should be made on other types of and larger scale distributed constraint programming problems.