

```

$ lisp
> (require :lcc)
> (in-package :lcc)
> (load "frameworks/astrogrid")
> (?- (run-experiment astrogrid random-incremental 5.0 8.0 1.0 10))

```

The first two lines load the LCC code and switch to the package. The third line loads a protocol framework and the relevant agent code. The final line calls the logic predicate `run-experiment/6` on the protocol `astrogrid`. The second argument specifies the matchmaking policy to use: policies available in this implementation are `random-incremental`, `ic-incremental`, and `ic-joint`. The latter two correspond to the algorithms in 1. The third, fourth and fifth arguments are the interval and step size over which the experiment runs. These alter a variable which influences the likely success of the agents in performing the protocol. For the Astrogrid protocol, this changes the size of data file we deal with: larger files lead to greater sensitivity to selection. The final argument is the number of protocol invocations to make.

---

Figure 1: Algorithms

IC-JOINT(*protocol*, *database*)

```

1  roles ← ROLES-REQUIRED(protocol)
2  collaborations ← ALL-COLLABORATIONS(protocol, database, roles)
3  for c ∈ collaborations
4      do quality[c] ← PROBABILITY-GOOD-OUTCOME(protocol, database, c)
5  return ARG-MAX(collaborations, quality)

```

IC-INCREMENTAL(*protocol*, *database*, *role*)

```

1  for r ∈ ACTIVE-ROLES(protocol)
2      do collaborators[r] ← COLLABORATOR-FOR-ROLE(protocol, r)
3  candidates ← CAPABLE-AGENTS(database, role)
4  for c ∈ candidates
5      do collaborators[role] ← c
6      quality[a] ← PROBABILITY-GOOD-OUTCOME(database, collaborators)
7  return ARG-MAX(candidates, quality)

```

ARG-MAX as used here does not always select the highest value. To improve the exploration of options, those entries that have low numbers of data points (i.e. have not often been selected previously) are preferentially chosen, and in other cases a random selection is sometimes made.

---