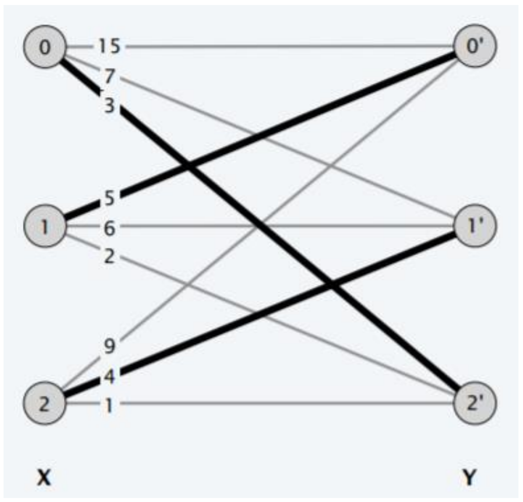
# Linear Assignment

Functions: linear algebra, matrix math
Inputs: cost (edge weight) matrices
Metrics: compute time

## 1. Overview

Linear assignment is a combinatorial optimization problem about finding the "best" mapping between two sets. Mathematically speaking, this means minimizing the sum of the edge weights in a bipartite graph. Practically speaking, this means finding the most efficient mapping, say, between a set of tasks and a set of agents that can perform those tasks, based on each agent's suitability for the various tasks. For instance, if one needs to use a swarm of drones with heterogeneous capabilities to perform distributed mapping, package delivery, *etc.*, linear assignment principles can help determine the optimal method of dividing sub-tasks among members of the swarm.



In this assignment problem, the optimal (minimum-cost) complete mapping is indicated in bold. Source: https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/07NetworkFlowIII-2x2.pdf

MilSpec's linear assignment benchmark includes two input modes and three selectable algorithms, all specified in testcase configuration files. The matrix input mode uses a predefined cost (edge weight) matrix, while the random input mode generates a matrix (of prespecified size) of random costs. The three algorithm implementations – auction, Hungarian, and Jonker-Volgenant – are derived from assorted open-source implementations. The auction algorithm[1] "operates like an auction whereby unassigned persons bid simultaneously for objects,"[2] where, e.g., the persons and objects are agents and tasks, respectively, and the bids relate to the performance of an agent at a given task. The Hungarian algorithm[3] iteratively finds an optimal solution by determining the lowest-cost edges from each vertex in one set and then progressively

[1] https://github.com/bkj/cuda_auction
[2] Bertsekas, D.P. "The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem." MIT, March 1987. http://www.columbia.edu/~cs2035/courses/ieor8100.F12/auction-alg.pdf
[3] https://github.com/mcximing/hungarian-algorithm-cpp

evaluating the next-lowest-cost edges (with respect to the entire set) until a complete mapping is found.[4] The Jonker-Volgenant algorithm[5] is a refinement of the Hungarian algorithm that runs in $O(n^3)$ instead of the original Hungarian algorithm's $O(n^4)$.[6] For MilSpec, these three algorithms/implementations were translated from their original implementations into C/C++ modules with a common interface, then integrated into a single testharness with performance measurement for streamlined testing/comparison.

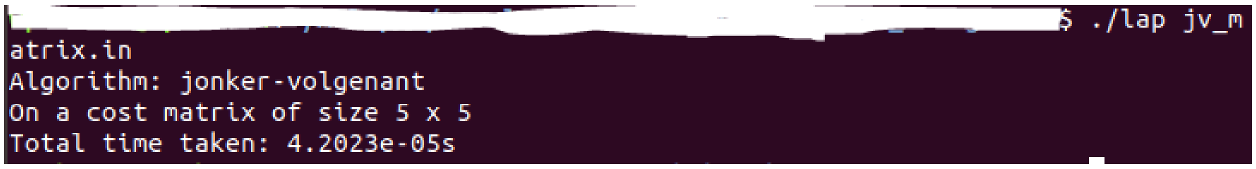
## 2. System requirements

Platform: Ubuntu 18.04 LTS with g++ 7.4.0. Code may build and run successfully on other versions/platforms, but has not been tested with them.
Storage: ~1MB for code and sample inputs. RAM requirements vary significantly by testcase, from ~1MB for small input matrices to ~4GB for large randomly-generated inputs.
Dependencies: None.

## 3. Build and run

To benchmark:

- Download and extract the zipfile from www.adacenter.org/milspec
- From the `LinearAssignment/` directory, `make clean && make`
- Choose a testcase. Six examples are provided, one for each combination of [algorithm, input mode]. For instance, for a test with a predefined edge weight matrix using the Jonker-Volgenant algorithm, use `jv_matrix.in`.
- From the `LinearAssignment/` directory, `./lap [testcase]`
- Results are displayed in the terminal, as below:



```
[redacted]$ ./lap jv_m
atrix.in
Algorithm: jonker-volgenant
On a cost matrix of size 5 x 5
Total time taken: 4.2023e-05s
```

## 4. Code structure

Coming soon!

## 5. MilSpec development notes, errata, changelog

v0.9:

- Changes from baseline algorithm code for MilSpec:
    - Ported auction algorithm implementation from CUDA to plain-C

[4] For a more detailed description, see "The Hungarian Algorithm for the Assignment Problem," http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/Assignment/algorithm.html
[5] https://github.com/yongyanghz/LAPJV-algorithm-c
[6] Jonker, R. and Volgenant, A. "A shortest augmenting path algorithm for dense and sparse linear assignment problems." *Computing* 38, 325-340 (1987). https://link.springer.com/article/10.1007%2FBF02278710

- Edited Hungarian and Jonker-Volgenant code structure
- Created testharness to permit straightforward algorithm selection
- Added basic configuration options and file-based testcase input
- Added performance metrics
- Created makefile
- Created basic testcases

## 6. Acknowledgements

MilSpec is a project under development at the University of Michigan – Ann Arbor by Pete Ehrett, Bing Schaefer, Adrian Berding, Nathan Block, John Paul Koenig, Pranav Srinivasan, Todd Austin, and Valeria Bertacco. This project is supported by the Applications Driving Architectures Center, one of six centers of JUMP, a Semiconductor Research Corporation program co-sponsored by DARPA.

The original linear assignment algorithm implementations upon which the MilSpec versions are based are available at https://github.com/bkj/cuda_auction (auction), https://github.com/mcximing/hungarian-algorithm-cpp (Hungarian), and https://github.com/yongyanghz/LAPJV-algorithm-c (Jonker-Volgenant).