# Resource Allocation Algorithms
## Lecture 01

Lecture by Dr. Tami Tamir
Typeset by Steven Karas

2018-03-13
Last edited 18:12:40 2018-03-13

**Disclaimer** These lecture notes are based on the lecture for the course Resource Allocation Algorithms, taught by Dr. Tami Tamir at IDC Herzliyah in the spring semester of 2017/2018. Sections may be based on the lecture slides written by Dr. Tami Tamir.

# 1 Course Admin

Office hours will be held on Tuesdays between 14:30-15:30 in C118.

Course materials will be posted to the website.

Homework will make up 25% of the grade, and the final exam 75%. There will be approximately 6 exercises, and the lowest grade will be dropped. Homework should be completed on your own, however you can discuss the homework in groups (with attribution). Each homework will take on average 5-10 hours.

Some of the slides may be used for the first time.

**Topics** This course will focus on algorithms that model resource allocation, and how to evaluate such algorithms.

- NP Approximations

- Performance evaluation

- Scheduling Theory

- Facility Location

- Packing Problems

- Max flow - extensions

- Algorithmic Game Theory

# 2 $\mathcal{NP}$-completeness

Established by Cook in 1971[1]. $\mathcal{NP}$ is a complexity class of problems whose solutions can be checked in polynomial time. It is known that $\mathcal{P}$ is a subset of $\mathcal{NP}$. We do not yet know if $\mathcal{NP}$ is equal to $\mathcal{P}$, or if $\mathcal{P}$ is a strict subset of $\mathcal{NP}$.

$\mathcal{NP}$-complete problems are $\mathcal{NP}$-hard and also in $\mathcal{NP}$. $\mathcal{NP}$-hard problems are at least as hard as any problem in $\mathcal{NP}$. Formally, a problem $P$ is $\mathcal{NP}$-hard if all problems $R \in \mathcal{NP}$ are polynomially reducible to $P$: $R \leq_p P, \forall R \in \mathcal{NP}$. To show that a problem is $\mathcal{NP}$-complete, start by showing that the problem is in $\mathcal{NP}$, and then show a reduction of any known $\mathcal{NP}$-complete problem to that problem.

## 2.1 Review: Reductions

A problem $P$ is reducible to $Q$, denoted by $P \leq_p Q$, if given an algorithm for solving $Q$, we can use this solution to solve $P$ with complexity $p$. A problem is polynomially reducible if such a reduction can be shown to have polynomial complexity.

## 2.2 SAT

SAT is $\mathcal{NP}$-complete, as proven by Cook. The SAT problem is given a boolean expression with n variables, can we assign values such that the expression is true.

$$((x_1 \wedge x_2) \vee \neg((\neg x_1 \wedge x_3) \vee x_4)) \wedge \neg x_2$$

The proof was argued from first principles, not as a reduction.

## 2.3 k-clique

A clique in a graph $G$ is a subset of vertices that are fully connected to each other. The decision problem of whether there is a clique of size $k$ is called the K-CLIQUE problem.

**Proof of NP-completeness** 3-CNF is also $\mathcal{NP}$-complete, so we'll show that by solving K-CLIQUE, you can solve 3-CNF.

Create a graph such that each vertex represents the members of each clause, and create edges to all other non-contradictory vertices from other clauses. Find a clique of size $k$ where $k$ is the number of clauses in 3-CNF.

## 2.4 Traveling Salesman Problem

The TRAVELING SALESMAN PROBLEM is a $\mathcal{NP}$-complete problem. The proof follows from a reduction from HAMILTONIAN CYCLE to TSP.

There are many other $\mathcal{NP}$-complete problems that are well known.

# 3 Game Theory

A subfield of mathematics[1] concerned with the construction and analysis of models describing situations where decision making is required.

**Rock, Paper, Scissors** In the game rock paper scissors for two players, each player has 3 possible strategies. We can present the outcomes of this game using a matrix of possible strategies and their relative payoff:

|          | Rock | Paper | Scissors |
|----------|------|-------|----------|
| Rock     | 0,0  | -1,1  | 1,-1     |
| Paper    | 1,-1 | 0,0   | -1,1     |
| Scissors | -1,1 | 1,-1  | 0,0      |

A binary game is one in which there are exactly two possible strategies for each player.

## 3.1 Dominant strategies

A dominant strategy is a strategy that offers a higher payoff than any other, irrespective of the other players' strategies.

For example, take the prisoner's dilemma:

|               | Confess | Remain Silent |
|---------------|---------|---------------|
| Confess       | 5,5     | 0,20          |
| Remain Silent | 20,0    | 1,1           |

In this game, the dominant strategy is to confess, as it always results in a better outcome (5 years instead of 20, or 0 years instead of 1).

## 3.2 Nash Equilibrium

A Nash equilibrium is a steady state wherein all players are no better off changing their strategies unilaterally. This can hold even when there is a global optimum that is better than the current equilibrium.

In the prisoner's dilemma, both prisoner's confessing is the only Nash equilibrium for this game, even though both staying silent would provide them with a better outcome.

---

[1]Originally from economics

**Example: Asymmetric players**  In this example, there are two Nash equilibriums, despite being asymmetric.

|          | Football | Movie |
|----------|----------|-------|
| Football | 5,8      | 2,2   |
| Movie    | 2,2      | 8,5   |

## 3.3  Mixed strategies

A mixed strategy for a player in a game with $m$ strategies is a vector $p_1, \ldots, p_m$ such that $\sum_i p_i = 1$. $p_i$ is the probability for that player to select strategy $i$.

It has been proven that every game has a mixed Nash equilibrium.

**Example: Asymmetric players with disjoint strategies**  In this example, we consider a game which has no Nash equilibrium in pure strategies. The driver can either pay for parking, or not. The inspector can inspect or not.

|           | Inspect  | Don't Inspect |
|-----------|----------|---------------|
| Pay       | -10,5    | -10,10        |
| Don't pay | -100,90  | 0,-5          |

The mixed strategies $(0.95, 0.05)$ and $(0.1, 0.9)$ form a mixed Nash equilibrium for this game.

Let $(x, 1-x)$ be the optimal mixed strategy for the driver, and $(y, 1-y)$ be the optimal mixed strategy for the inspector.

To find the optimal strategy of the driver, we want to discover the point where the payoff is independent of the inspector:

$$5x + 90(1 - x) = 10x - 5(1 - x)$$
$$x = 0.95$$

Similarly, we can find the optimal strategy of the inspector:

$$-10y - 10(1 - y) = -100y$$
$$y = 0.1$$

The expected income of the driver is $-10$, and the expected profit of the inspector is $9.25$[2].

**Example: Chicken**  Two players driving towards each other. They can either swerve or drive.

|        | Swerve | Drive  |
|--------|--------|--------|
| Swerve | 1,1    | -1,2   |
| Drive  | 2,-1   | -M,-M  |

Both Swerve/Drive and Drive/Swerve are pure equilibriums[3].

# 4  Resource Allocation Games

A resource allocation game is comprised of a set of resources, a set of players, wherein each player needs a subset of resources to fulfill their needs. There may be several such subsets, which define the strategy space of the player.

In traditional resource allocation, a central utility assigns players to resources. In algorithmic game theory, each player is a selfish agent who selects their own assignments.

**Network Formation**  See the slides, this is more graphical than I can copy down.

Basically, the cost of establishing a new communication channel as part of the network is shared between all players who use that channel.

In such games, there are three interesting questions:

1. Is there a pure Nash equilibrium?

2. Can such an equilibrium be computed efficiently?

3. What is the quality of the equilibrium? This is typically the "social cost" as measured by the total cost to all players.

---

[2] A full explanation of this expectation will be uploaded to the course website
[3] A full explanation of a mixed Nash equilibrium will be uploaded to the course website

**Golden Balls**  As an real life example of the prisoner dilemma, there was a British game show that showed players who could split 100K GBP or steal the full amount.

# References

[1] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.