

# Workshop 2: Approximation Algorithms

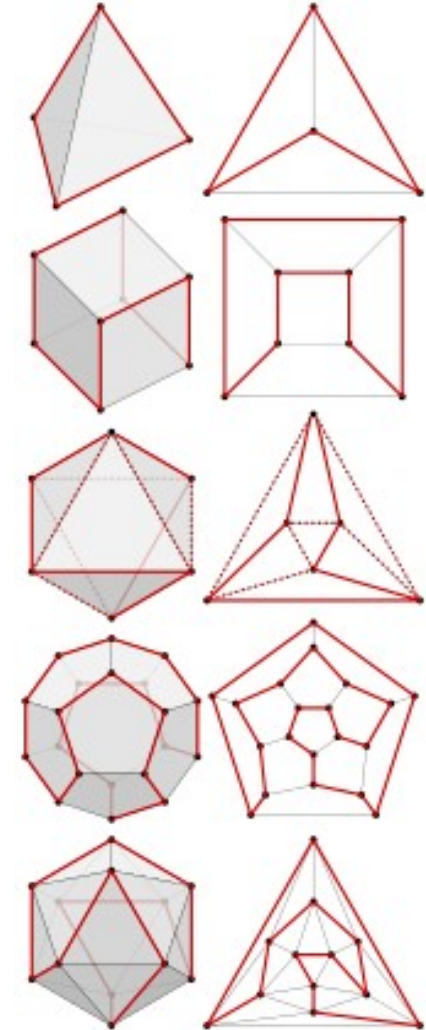
CITS3001 Algorithms, Agents and Artificial  
Intelligence

A TABLE OF  
SLIGHTLY WRONG  
EQUATIONS AND IDENTITIES  
USEFUL FOR  
APPROXIMATIONS  
AND/OR  
TROLLING TEACHERS  
(FOUND USING A MIX OF TRIAL-AND-ERROR,  
MATEMATICA, AND ROBERT MUNARO'S PIES TOOL.)  
ALL UNITS ARE SI MKS UNLESS OTHERWISE NOTED.

RELATION:		ACCURATE TO WITHIN:
ONE LIGHT-YEAR(m)	$99^8$	ONE PART IN 40
EARTH SURFACE(m <sup>2</sup> )	$69^8$	ONE PART IN 130
OCEANS VOLUME(m <sup>3</sup> )	$91^9$	ONE PART IN 70
SECONDS IN A YEAR	$75^4$	ONE PART IN 400
SECONDS IN A YEAR (RENT METHOD)	$525,600 \cdot 60$	ONE PART IN 1400
AGE OF THE UNIVERSE (SECONDS)	$15^{15}$	ONE PART IN 70
PLANCK'S CONSTANT	$\frac{1}{30\pi^e}$	ONE PART IN 110
FINE STRUCTURE CONSTANT	$\frac{1}{140}$	[I'VE HAD ENOUGH OF THIS 137 CONST]
FUNDAMENTAL CHARGE	$\frac{3}{14\pi\pi^{\pi}}$	ONE PART IN 500
WHITE HOUSE SWITCHBOARD	$e^{\frac{1}{\sqrt{1+c\sqrt{8}}}}$	
JENNY'S CONSTANT	$(7^{\frac{9}{2}} - 9)\pi^{\pi}$	

# An NP complete problem

- A Hamiltonian Cycle is a path through a graph that starts and ends at the same vertex, and visits every other vertex of the graph exactly once.
- Consider the problem of given a graph, finding a Hamiltonian Cycle, if one exists.
- Is this problem easier or harder than the travelling salesmen problem?
- An algorithm is *correct* if when it returns, it always gives the right answer, and is *complete* if it always returns.
- Describe a correct and complete algorithm for this problem?
- What is the most efficient correct and complete method for solving this problem?
- Describe a correct, but not complete, algorithm to solve this problem.



# State space search

- Suppose we have a 0-1 knapsack problem, with  $n$  items, each with a weight and a value, and a knapsack of size  $W$ .
- How would we express the solution space of this problem?
- How big is the solution space?
- What would a suitable neighbourhood function be?
- What are some search strategies that could be used? Are there any clever optimisations we can use? Are there any breaking cases where our search might fail?



# Genetic Algorithms

- A genetic algorithm uses a population of candidate solutions, a mechanism for mutating and combining (or breeding new solutions) and the principle of survival of the fittest to evolve a near optimal solution.
- Use these principles to build a genetic algorithm to solve the 0-1 knapsack problem. The key parts are:
  - A *genome*, which is a vector that represents a single solution
  - A method to *mutate* and *crossover* the genome
  - A *fitness function*, and
  - A *selection mechanism* to determine which solutions survive.
- Try implementing this algorithm and compare the results to your algorithm for Lab 2.

