LA TROBE UNIVERSITY

All kinds of clever

Prescriptive Analytics – From Conventional Methods to AI-assisted Problem Solving

AAI Autumn School

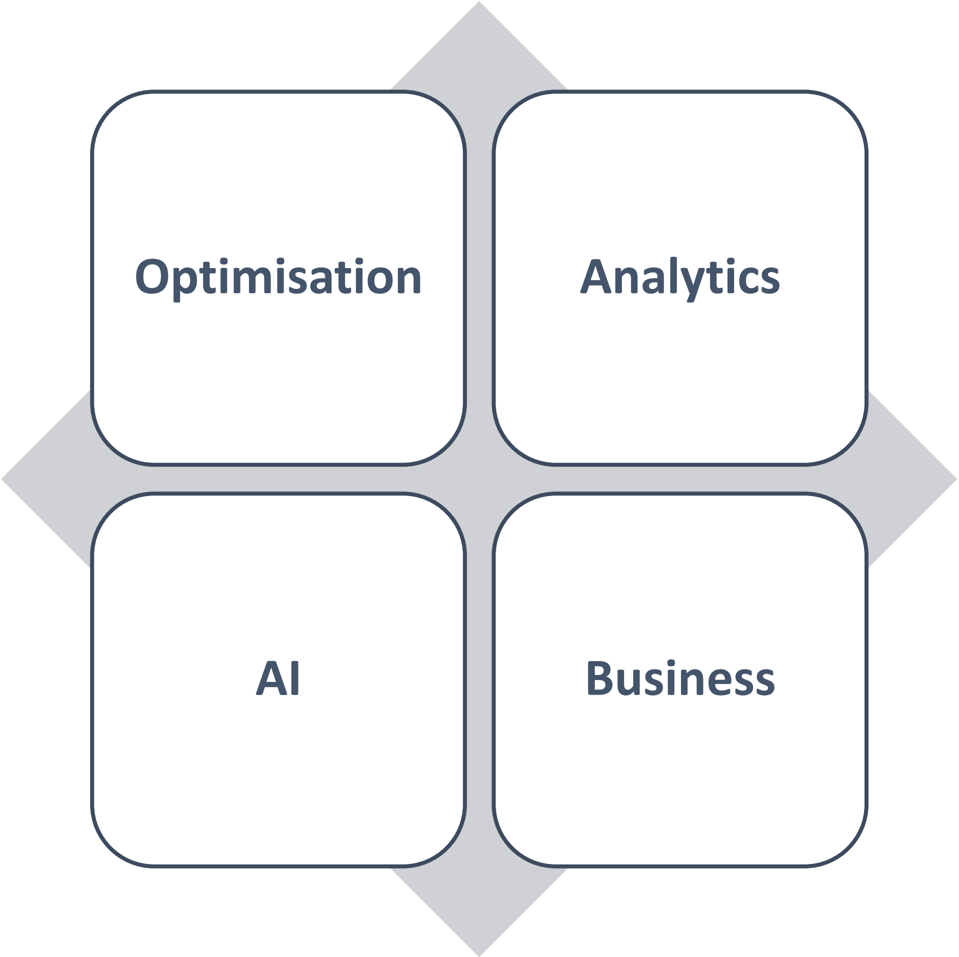
Su Nguyen

Centre for Data Analytics and Cognition

La Trobe Business School (LBS)

Outline

- Prescriptive Analytics
- Optimisation problems and solution methods
- Issues and challenges
- AI-assisted problem solving
  - Machine learning
  - Evolutionary learning
- Future research directions



IBM Prescriptive Analytics

Helps organizations make better decisions by solving complex optimization problems involving trade-offs between business goals and constraints

Prescriptive Analytics is a form of advanced analytics which examines data or content to answer the question "What should be done?" or "What can we do to make _______ happen?"

Gartner

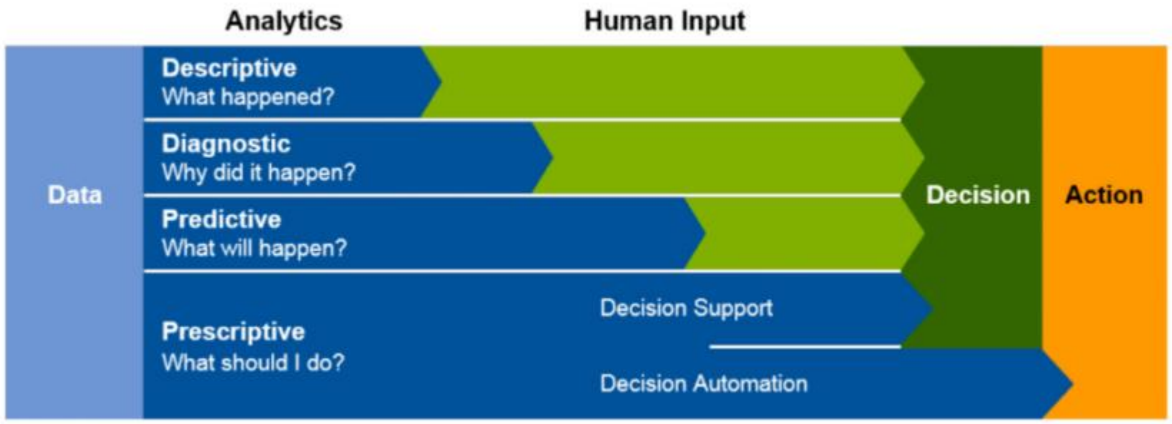
Prescriptive analytics […] are analytics that tell you what to do. They make recommendations—often to front-line workers—about the best way to handle a given situation. What price to charge for a product, what version of a webpage to use, what turn to make next in a driving route—all of these are prescriptive analytics.

Thomas H. Davenport

(in The Rise of Automated Analytics, WSJ, 2015)

Prescriptive Analytics

- Prescriptive Analytics is the third and final phase of business analytics
- Prescriptive Analytics is more problem centric than descriptive analytics and predictive analytics
  - What should I do?
  - What should I do that?
  - What are the objectives/preferences?
  - What are the constraints/boundaries?
  - etc.
- Prescriptive Analytics uses knowledge from a wide range of scientific discipline, especially Operations Research



Source: Gartner (May 2015)

Operations Research – Early Brand of Prescriptive Analytics

John von Neumann
(Game Theory)

George B. Dantzig
(Linear Programming)

1881-1935 (Pre OR)
- Scientific management
- Motion studies
- Probability and statistics

1936-1946 (Birth of OR)
- Military applications
- Basic computer
- TSP
- Game theory

1947-1950 (Expansion of OR)
- Linear programming
- Statistical quality control
- Dynamic programming

1951-1956 (Development of OR tools)
- Computer based optimization
- Inventory Control
- Sequencing and Scheduling
- OR in university

1957-1963 (Applications and Activities)
- First conference
- More applications
- New theories
- OR textbooks
- Simulation

1964-1990
- More theories
- Less focus on empirically-oriented research
- Meta-heuristics

Richard Bellman
(Dynamic Programming)

Herbert A. Simon
(Bounded Rationality)

John Holland
(Genetic Algorithm)

Until now?
- Focus more on empirical studies
- Incorporate new technologies
- Data-driven optimization
- Adapt to dynamic and uncertainty environments

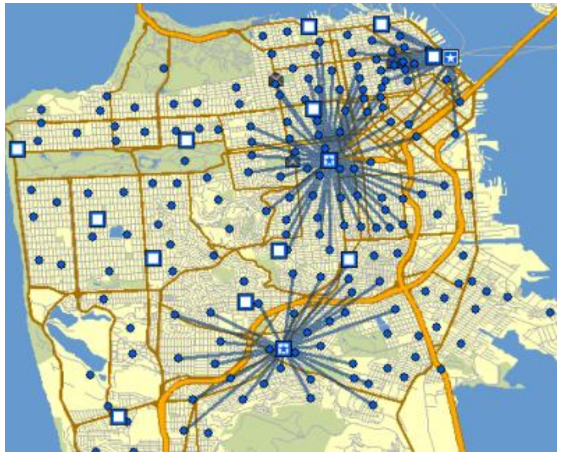
Optimisation (1)

- **Mathematical optimization or mathematical programming** is the selection of a best element (with regard to some criterion) from some set of available alternatives. ["The Nature of Mathematical Programming Archived 2014-03-05 at the Wayback Machine," Mathematical Programming Glossary, INFORMS Computing Society.]

2	7	3
6	1	5
4	8	9

Find the 3 cells that have the maximum sum and they are not in the same rows or columns

$$minimise\ f(x) = x^2 + 1$$
$$minimise\ f(x, y) = x^2 + y^2$$



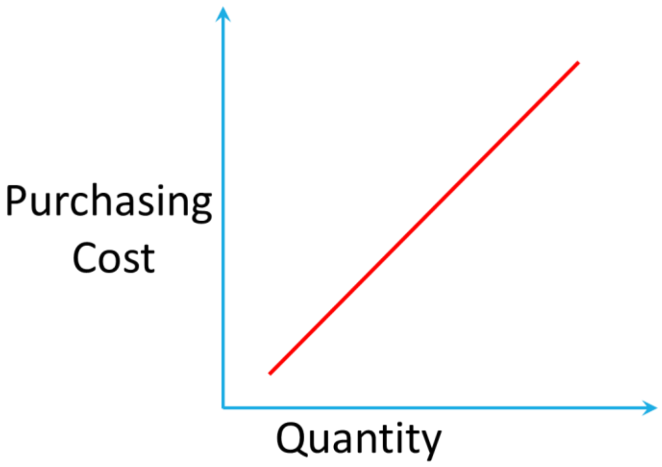
Choosing the facility locations to minimise delivery times and costs

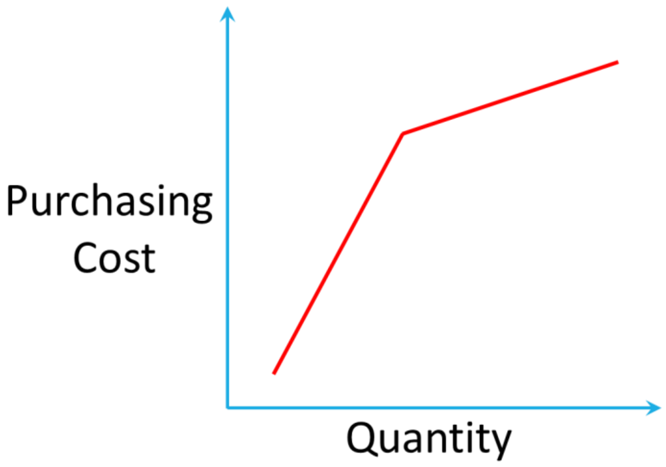
Optimisation (2)

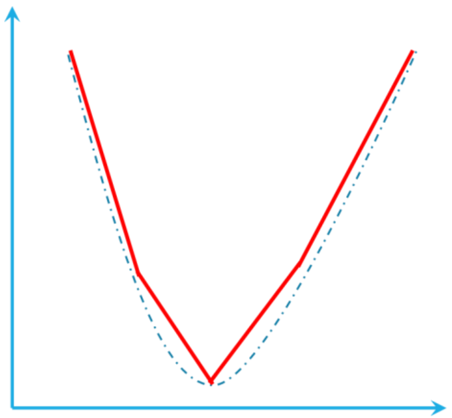
- We only focus on optimisation problems that can be solved by computer algorithms.
- In this case, we need to transform the optimisation problems in the formats that can be solved by computers
  - Mathematical formulation
  - Simulation models
  - Data-driven
- To facilitate this process, we need to introduce several concepts:
  - Input data
  - Decision variables
  - Constraints
  - Objective function

Linear Programming (LP)

- LP is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships.
- Why are we interested in linear relationships?
- Well … many useful things in our daily life have linear relationships.

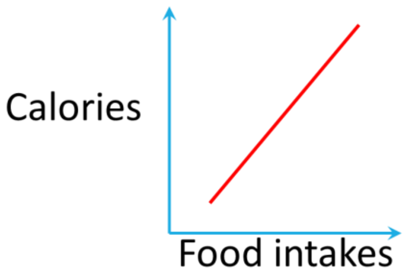






Example 1: Diet Problem

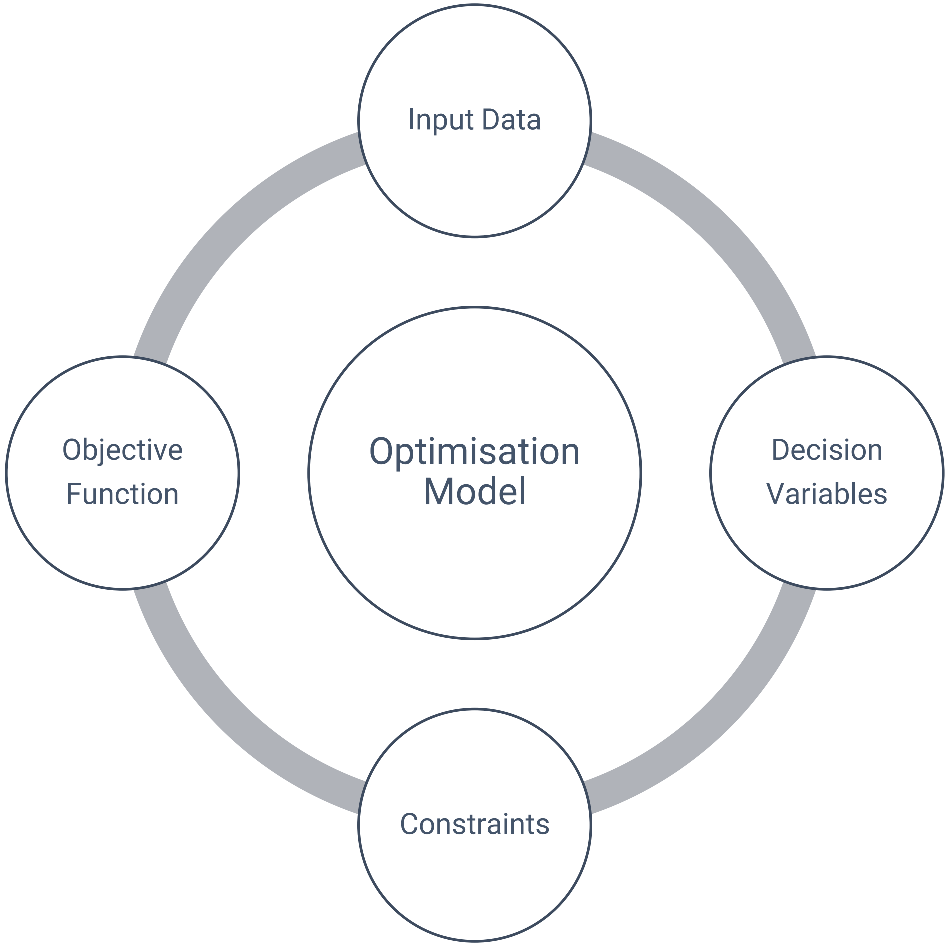
- Consider the following list of foods with their nutritional profile:



		Calories (Kcal)	Fat (g)	Saturated fat (g)	Trans fat (g)	Cholesterol (mg)	Protein (g)	Vitamin A (IU)	Vitamin C (mg)	Calcium (mg)	Iron (mg)	
Nutritional bounds	Min	1900	30	0	0	0	50	3000	300	1000	18	
on optimized diet	Max	2200	75	25	2	300	160	7000			40	
Food	**Serving size**											**Cost per Portion ($)**
Donut	60g	239	11	3	0	18	4	13.8	0.7	27.6	2.2	0.85
Bagel, Oat Bran	57g	145	1	0	0	0	6	2.3	0.1	0.8	1.8	0.85
Yogurt,fruit,non fat	125g	119	0	0	0	2	6	15	0.9	190	0.1	1.25
Chili	216g	190	8	3	0	25	14	1250	36	80	1.4	2.5
Brocolli,boiled,no salt	140g	49	1	0	0	0	3	2167	90.9	56	0.9	0.4
Apple	182g	95	0	0	0	0	0	98.3	8.4	10.9	0.2	0.4
Oats,instant,dry	328g	105	2	0	0	0	4	1000	0	98.5	8.2	0.5
Orange,raw,navel	140g	69	0	0	0	0	1	346	82.8	60.2	0.2	0.3
Lentils,cooked,no salt	100g	116	0	0	0	0	9	8	1.5	19	3.3	0.35
Carrots, baby raw	28g	10	0	0	0	0	0	3861	0.7	9	0.2	0.3
Brussel sprout, cooked	78g	28	0	0	0	0	2	604	48.4	28.1	0.9	0.5
Chicken, roast, no skin	140g	234	9	3	0	105	35	57.5	0	16.8	1.7	1.5
Blueberries, raw	28g	16	0	0	0	0	0	15.1	2.7	1.7	0.1	0.6
Spinach,boiled, no salt	180g	41	0	0	0	0	5	18870	17.6	245	6.4	0.3
Banana, raw	118g	105	0	0	0	0	1	75.5	10.3	5.9	0.3	0.3
Milk 1%, added vit A	244g	102	2	2	0	12	8	478	0	290	0.1	0.5

Goal: select a set, portions of foods that meets certain daily nutritional requirements and preferences and additionally at minimum cost.

Problem Formulation (1)



Problem Formulation (2)

- What are the inputs/parameters?
  - ✓ Types of foods
  - ✓ Serving size for each portion of food
  - ✓ Nutrition values for each type of food
  - ✓ Cost for each portion of food
  - ✓ Maximum/minimum nutrient requirements
  - ✓ Preferences: e.g. at least one portion of yogurt
- What are the decision variables?
  - ✓ Number of portions for each type of food
- What are the constraints?
  - ✓ Daily nutrition requirements
  - ✓ Preference constraint(s)
- What is the objective?
  - ✓ Minimise the total cost

Problem Formulation (3)

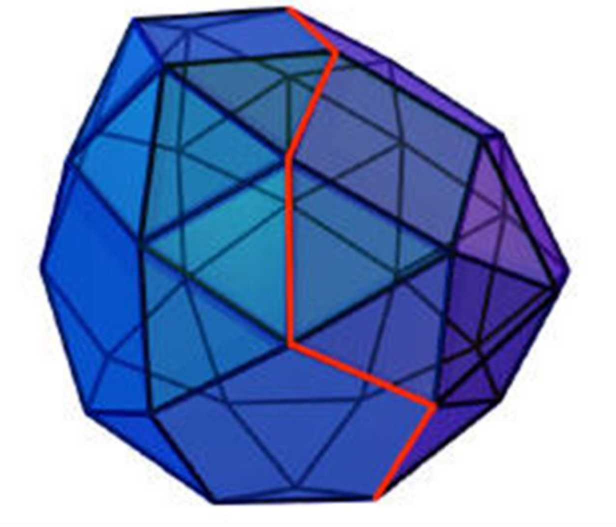
- As using the full name is a bit lengthy, we will assign some IDs to the foods and the nutrients.
  - ✓ 1,2,3,… for Donut, Bagel, Yogurt, … respectively
  - ✓ 1,2,3,… for Calories, Fat, Saturated fat, … respectively
- let use i as the index for food and j as the index for nutrient
  - ✓ x[i] is the number of portions of food i
  - ✓ c[i] is the cost for each portion of food i
  - ✓ a[i,j] is the amount of nutrients j in one portion of food i
- Daily requirement constraints:
  - ✓ x[1] * a[1,1] + x[2] * a[2,1] + x[3] * a[3,1] + … >= MIN_REQUIREMENT_OF_NUTRIENT_1 (i.e. Calories)
  - ✓ x[1] * a[1,1] + x[2] * a[2,1] + x[3] * a[3,1] + … <= MAX_REQUIREMENT_OF_NUTRIENT_1 (i.e. Calories)
- Objective: total cost = x[1]*c[1] + x[2]*c[2] + x[3]*c[3] + …

Problem Formulation (4)

- Linear Programming Model for the Diet Problem (N types of foods, M types of nutrients):
- **Objective**:
  - Minimise $\sum_{i=1}^{N} c[i]x[i]$
- **Constraints**:
  - $\sum_{i=1}^{N} a[i,j]x[i] \leq MAX\_REQUIREMENT\_OF\_NUTRIENT\_j$ for all nutrient j = 1…M
  - $\sum_{i=1}^{N} a[i,j]x[i] \geq MIN\_REQUIREMENT\_OF\_NUTRIENT\_j$ for all nutrient j = 1…M
  - $x[i] \geq 0$ for all food i = 1…N

Solvers for LP

- **Excel Solver**
  - For small problems
- **Open source softweare**
  - CBC-CoinOR
  - GLPK
  - Google OR-tools
  - Cxvpy
- **Commercial**
  - IBM Ilog Cplex
  - Gurobi
  - XPress

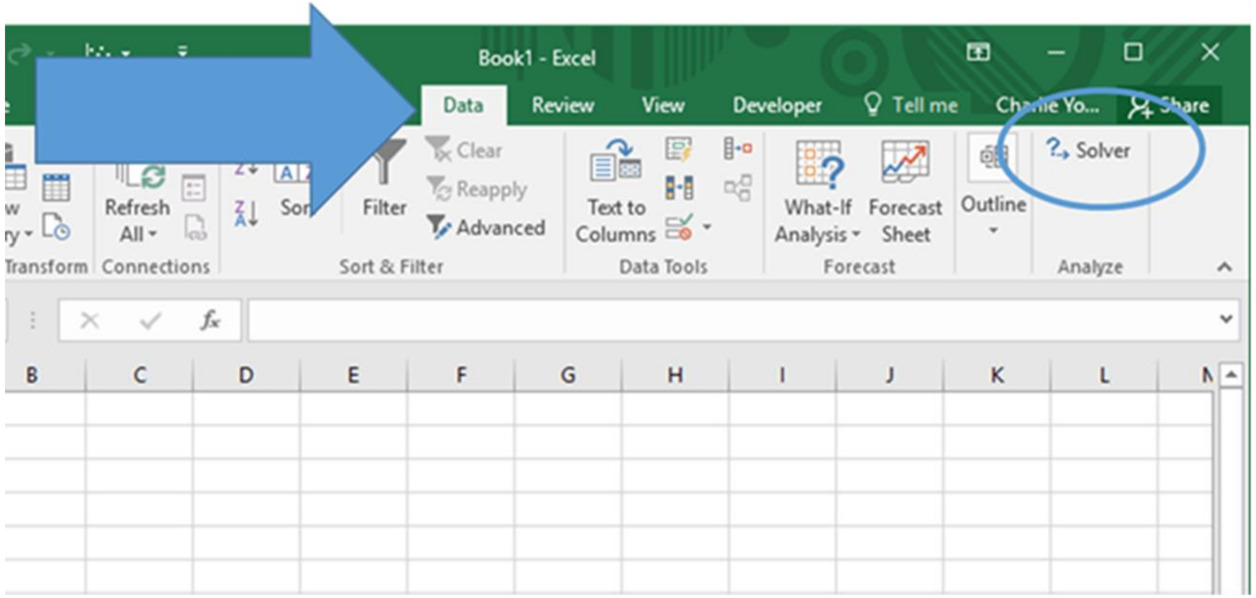


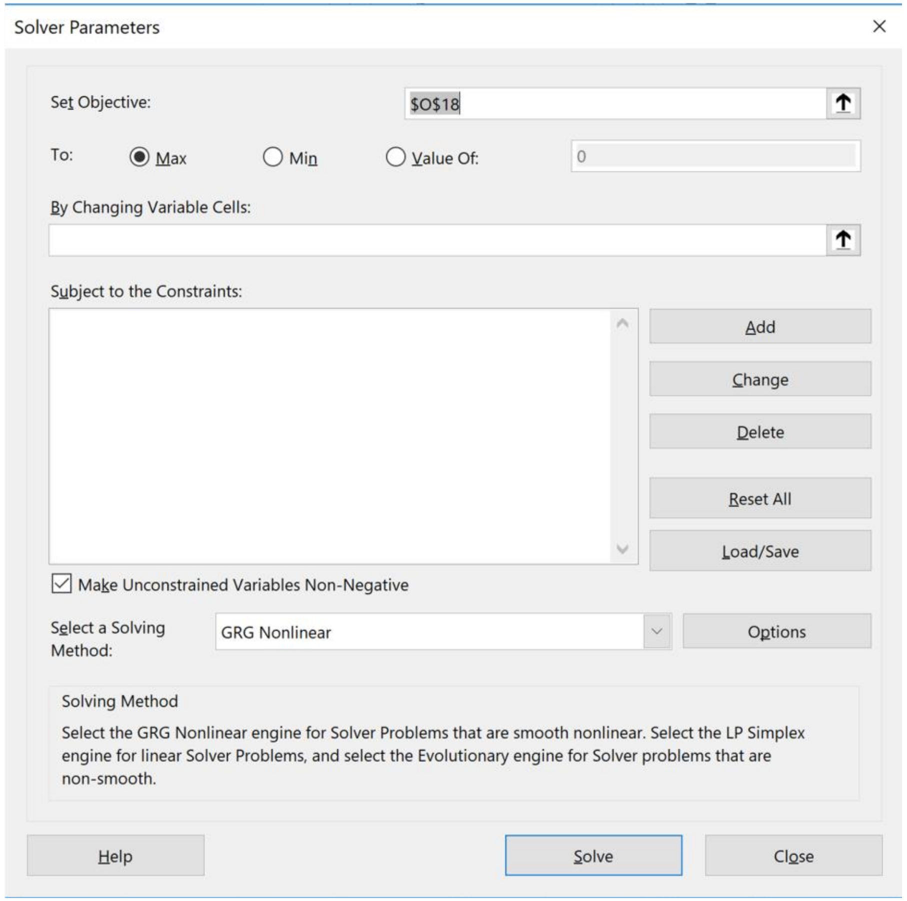
https://softmath.com/tutorials-3/cramer's-rule/exact-computation-of-basic.html

Excel Solver (1)

- Useful and available as an add-in in MS Excel (activate through File>Options>Add-ins)
  - There are a number of solvers to deal with different optimisation problems
  - Simplex for LP problems
  - GRG for smooth non-linear problems
  - Evolutionary Algorithms to deal with non-smooth problems
- Pros:
  - Easy to formulate simple problems if the data is well-organised in Excel worksheets
  - Able to handle integer/binary variables
- Cons:
  - Slow as compared to specialised optimisation software
  - Problem formulation becomes troublesome when the problem size increases

Excel Solver (2)





How to add Solver to Excel: https://support.microsoft.com/en-us/office/load-the-solver-add-in-in-excel-612926fc-d53b-46b4-872c-e24772f078ca

Solve Diet Problem with Excel Solver

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
11			Calories (Kcal)	Fat (g)	Saturated fat (g)	Trans fat (g)	Cholesterol (mg)	Sodium (mg)	Carbs (g)	Fibre (g)	Sugar (g)	Protein (g)	Vitamin A (IU)	Vitamin C (mg)	Calcium (mg)	Iron (mg)		
12	Nutritional bounds	Min	1900	30	0	0	0	1500	150	20	0	50	3000	300	1000	18		
13	on optimized diet	Max	2200	75	25	2	300	2300	300	50	50	160	7000	1E+10	1E+10	40		
14	Food	Serving size						Nutrition per portion									Cost per Portion ($)	Optimum Number of Portions
15	Donut	60g	239	11	3	0	18	232	30	1	12	4	13.8	0.7	27.6	2.2	0.85	
16	Bagel, Oat Bran	57g	145	1	0	0	0	289	30	2	1	6	2.3	0.1	0.8	1.8	0.85	
17	Yogurt,fruit,non fat	125g	119	0	0	0	2	72	24	0	24	6	15	0.9	190	0.1	1.25	
18	Chili	216g	190	8	3	0	25	1040	17	5	5	14	1250	36	80	1.4	2.5	
19	Brocolli,boiled,no salt	140g	49	1	0	0	0	57	10	5	2	3	2167	90.9	56	0.9	0.4	
20	Apple	182g	95	0	0	0	0	2	25	4	19	0	98.3	8.4	10.9	0.2	0.4	
21	Oats,instant,dry	328g	105	2	0	0	0	72	19	3	0	4	1000	0	98.5	8.2	0.5	
22	Orange,raw,navel	140g	69	0	0	0	0	1	18	3	12	1	346	82.8	60.2	0.2	0.3	
23	Lentils,cooked,no salt	100g	116	0	0	0	0	2	20	8	2	9	8	1.5	19	3.3	0.35	
24	Carrots, baby raw	28g	10	0	0	0	0	22	2	1	1	0	3861	0.7	9	0.2	0.3	
25	Brussel sprout, cooke	78g	28	0	0	0	0	16	6	2	1	2	604	48.4	28.1	0.9	0.5	
26	Chicken, roast, no skir	140g	234	9	3	0	105	105	0	0	0	35	57.5	0	16.8	1.7	1.5	
27	Blueberries, raw	28g	16	0	0	0	0	0	4	1	3	0	15.1	2.7	1.7	0.1	0.6	
28	Spinach,boiled, no sal	180g	41	0	0	0	0	126	7	4	1	5	18870	17.6	245	6.4	0.3	
29	Banana, raw	118g	105	0	0	0	0	1	27	3	14	1	75.5	10.3	5.9	0.3	0.3	
30	Milk 1%, added vit A	244g	102	2	2	0	12	107	13	0	13	8	478	0	290	0.1	0.5	
31	**Nutrition in Optimized Diet**		**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**		
32													TOTAL COST OF CHEAPEST DIET				0	
33																		
34																		
35																		

Constraints used in Solver

Cells to be changed by Solver

Cell to be minimized by Solver

The Diet Problem

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
2		Calories (Kcal)	Fat (g)	Saturated fat (g)	Trans fat (g)	Cholesterol (mg)	Sodium (mg)	Carbs (g)	Fibre (g)	Sugar (g)	Protein (g)	Vitamin A (IU)	Vitamin C (mg)	Calcium (mg)	Iron (mg)		
3	Nutritional bounds (MIN)	1900	30	0	0	0	1500	150	20	0	50	3000	300	1000	18		
4	on optimized diet (MAX)	2200	75	25	2	300	2300	300	50	50	160	7000	1E+10	1E+10	40		
5	Food	Nutrition per portion														Cost per Portion ($)	Optimum Number of Portions
6				3	0	18	232	30	1	12	4	13.8	0.7	27.6	2.2	0.85	0
				0	0	0	289	30	2	1	6	2.3	0.1	0.8	1.8	0.85	2.8504195
				0	0	2	72	24	0	24	6	15	0.9	190	0.1	1.25	0
				3	0	25	1040	17	5	5	14	1250	36	80	1.4	2.5	0
				0	0	0	57	10	5	2	3	2167	90.9	56	0.9	0.4	0.9874009
				0	0	0	2	25	4	19	0	98.3	8.4	10.9	0.2	0.4	0
				0	0	0	72	19	3	0	4	1000	0	98.5	8.2	0.5	2.1200316
				0	0	0	1	18	3	12	1	346	82.8	60.2	0.2	0.3	1.181398
				0	0	0	2	20	8	2	9	8	1.5	19	3.3	0.35	3.1272251
				0	0	0	22	2	1	1	0	3861	0.7	9	0.2	0.3	0
				0	0	0	16	6	2	1	2	604	48.4	28.1	0.9	0.5	2.2200336
				3	0	105	105	0	0	0	35	57.5	0	16.8	1.7	1.5	2.2758321
				0	0	0	0	4	1	3	0	15.1	2.7	1.7	0.1	0.6	0
				0	0	0	126	7	4	1	5	18870	17.6	245	6.4	0.3	0
				0	0	0	1	27	3	14	1	75.5	10.3	5.9	0.3	0.3	0
				2	0	12	107	13	0	13	8	478	0	290	0.1	0.5	1.7325784
				10.29265	0	259.753309	1500	255.32	50	50	155.826	7000	300	1000	40		
												TOTAL COST OF CHEAPEST DIET				10.716835	

Constraints used in Solver

Cells to be changed by Solver

Cell to be minimized by Solver

Solver Parameters

Set Objective: $P$23

To: ( ) Max (•) Min ( ) Value Of: 0

By Changing Variable Cells: $Q$6:$Q$21

Subject to the Constraints:

$B$22:$O$22 <= $B$4:$O$4
$B$22:$O$22 >= $B$3:$O$3

Add | Change | Delete | Reset All | Load/Save

☑ Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP | Options

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Help | Solve | Close

Go to **Data tab > Solver**

Add your preferences and resolve

- What is the optimal cost if I insist to have at least 1 Donut?
- Is it possible to include 5 Donuts in Diet?
- What is the optimal cost if I need at least 20mg of Iron?

Example 2: Economic lot-sizing

- **Case study:** The manager of a computer store is very happy with the sales of a new line of smart home device (SHD) which was launched last year. For planning purposes, he has asked the analytics team to build a predictive model to forecast the demand of the SHD in the next 10 weeks. The team has come up with a very accurate model and the predicted demand is shown in the below table:

Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8	Wk9	Wk10
42	42	32	12	26	112	45	14	76	38

- To get these items, the manager need to order from the distributors and it will cost $132 per order (or admin and transportation costs). The holding costs for each device per week is $0.6. The store needs to make sure that they have enough inventory to satisfy the demand in the next 10 weeks (assuming that their inventory is zero at Week 0) while minimise their total cost.
- This problem is referred to as the economic lot-sizing (ELS) problem

Economic lot-sizing – Simple Plan 1

- Place new order every week

Ordering Cost ($/order)	132
Holding cost ($/(unit) per week	0.6
Initital Inventory	0

B2

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Total
Demand	42	42	32	12	26	112	45	14	76	38	439
Order Quatity	42	42	32	12	26	112	45	14	76	38	439
Ordering Cost	132	132	132	132	132	132	132	132	132	132	1320
End-period inventory	0	0	0	0	0	0	0	0	0	0	0
Holding Cost	0	0	0	0	0	0	0	0	0	0	0
Total Cost (Order Cost + Holding Cost)											1320

B9

=IF(B9>0,$B$2,0)

Economic lot-sizing – Simple Plan 2

- One order to cover demand of 10 weeks

Ordering Cost ($/order)	132
Holding cost ($/(unit) per week	0.6
Initial Inventory	0

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Total
Demand	42	42	32	12	26	112	45	14	76	38	439
Order Quatity	439	0	0	0	0	0	0	0	0	0	439
Ordering Cost	132	0	0	0	0	0	0	0	0	0	132
End-period inventory	397	355	323	311	285	173	128	114	38	0	2124
Holding Cost	238.2	213	193.8	186.6	171	103.8	76.8	68.4	22.8	0	1274.4
Total Cost (Order Cost + Holding Cost)											1406.4

Economic lot-sizing – Simple Plan 3

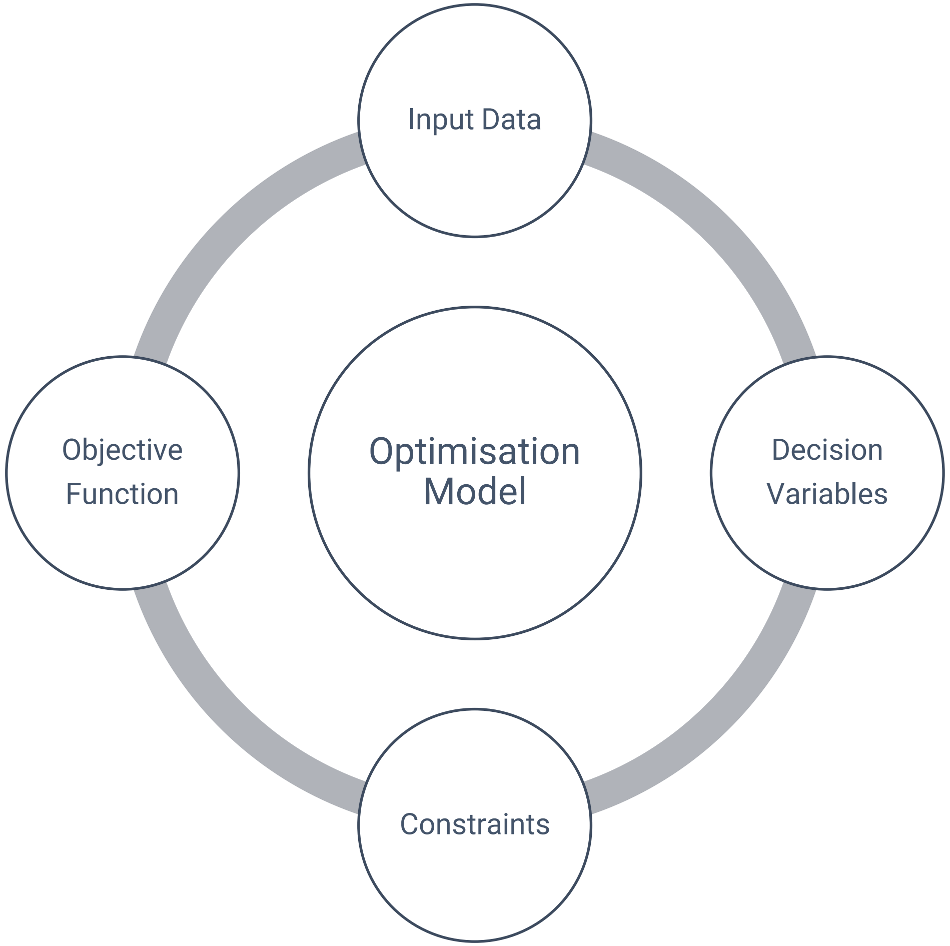
- Place order to cover the next two weeks

Ordering Cost ($/order)	132
Holding cost ($/(unit) per week	0.6
Initital Inventory	0

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Total
Demand	42	42	32	12	26	112	45	14	76	38	439
Order Quatity	84	0	44	0	138	0	59	0	114	0	439
Ordering Cost	132	0	132	0	132	0	132	0	132	0	660
End-period inventory	42	0	12	0	112	0	14	0	38	0	218
Holding Cost	25.2	0	7.2	0	67.2	0	8.4	0	22.8	0	130.8
Total Cost (Order Cost + Holding Cost)											790.8

Much better ~ 50% cost reduction!!! But is it the best?

Economic lot-sizing – LP approach (1)



Mathematical model of ELS

t: a period (e.g., day, week, month); $t = 1, \ldots, T$, where T represents the *planning horizon*
$\boldsymbol{D_t}$: demand in period t (number of units)
$\boldsymbol{c_t}$: unit purchasing/production cost
$\boldsymbol{A_t}$: ordering/setup cost associated with placing an order (or initiating production) in period t
$\boldsymbol{h_t}$: cost of holding one unit of inventory from period t to period $t+1$
$\boldsymbol{Q_t}$: the size of the order (or lot size) in period t; a decision variable
$\boldsymbol{I_t}$: the inventory level at the end of period t
$\boldsymbol{Y_t}$: ordering decision in period t (0 for no, 1 for yes)

$$\text{Minimize} \quad z = \sum_{t=1}^{T} c_t Q_t + h_t I_t + A_t Y_t$$

subject to

$$
\begin{aligned}
I_t &= I_{t-1} + Q_t - D_t, && t = 1, \ldots, T \\
Q_t &\le Y_t \left(\sum_{t'=t}^{T} D_{t'}\right) && t = 1, \ldots, T \\
Y_t &\in \{0,1\} && t = 1, \ldots, T \\
Q_t, I_t &\ge 0 && t = 1, \ldots., T
\end{aligned}
$$

Economic lot-sizing – LP approach (2)

	A	B
2	Ordering Cost ($/order)	132
3	Holding cost ($/(unit) per week	0.6
4	Initital Inventory	0

Optimal solution: *order at Week 1, Week 6, and Week 9*
The optimal cost is **610.2**!!!

	A	B	C	D	E	F	G	H	I	J	K	L
44		Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Total
45	Demand	42	42	32	12	26	112	45	14	76	38	439
46	Order Quatity	154	0	0	0	0	171	0	1.78E-15	114	0	439
47	Ordering Decision (0: no, 1: yes)	1	0	0	0	0	1	0	0	1	0	
48	Ordering Cost	132	0	0	0	0	132	0	0	132	0	**396**
49	End-period inventory	112	70	38	26	0	59	14	0	38	0	357
50	Holding Cost	67.2	42	22.8	15.6	0	35.4	8.4	0	22.8	0	**214.2**
51	Total Cost (Order Cost + Holding Cost)											**610.2**
52												
53	*Dummy Variable (ordering decision * very large number*	*439*	*0*	*0*	*0*	*0*	*439*	*0*	*0*	*439*	*0*	

Other examples: Budget Allocation in Online Advertising

- An online advertising agency wishes to optimize conversions (sales) for an online retailer which wants to advertise a product.
- The campaign involves placing display ads over two sites, and giving a coupon to every person who clicks on the ad.
- The retailer is interested in maximizing the number of conversions.

Data – Online Advertisement

	CPI	CTR	Conversions	Trac
Site A	$0.01	0.02	0.003	4,000,000
Site B	$0.01	0.01	0.002	∞
Budget	80,000	100,000		

See Site (A or B) → Click to get coupon → Converted

Marketing performance data table. The first column shows that advertising on each site has a **Cost Per Impression (CPI) of 1 cent**. The **budget** allocated for the campaign is **$80,000**. The second column shows the **Click Through Rate (CTR)** on each site. The CTR is the ratio between the number of people who see the ad and the number who click on it. The retailer is willing to spend **100,000 coupons** on this campaign, in expectation. The third column shows the **conversion** rate on each site: the ratio between number of people who are shown the ad and those who end up making a purchase. The last column shows the **trac**: the number of people who visit the site in the time period which we intend to run the campaign.

Objective: Total Conversion (to maximise)
0.003*x1 + 0.002*x2

Where x1 and x2 are the number of impressions for Site A and Site B respectively

Budget constraint: 0.01*x1 + 0.01*x2 <= 80,000

Coupon constraint: 0.02*x1 + 0.01*x2 <= 100,000

Visit constraint: x1 <= 4,000,000

$$
\begin{aligned}
\max\ & (3x_1 + 2x_2) \cdot 10^{-3} \\
s.t.\ & x_1 + x_2 \leq 8 \cdot 10^6 \\
& 2x_1 + x_2 \leq 10 \cdot 10^6 \\
& x_1 \leq 4 \cdot 10^6
\end{aligned}
$$

Marketing Optimisation

- Allocating budget for marketing campaigns including **Facebook, TV, SEO, Adwords**
- Maximise the return on investment (ROI)
- The **total budget is $1,000,000**
- Search Engine Marketing (SEO + Adwords) is the primary focus and spend must exceed 60% of the total budget
- Social media campaign on Facebook should cost no more than 20% of the budget.
- Production and airing of a TV ad will cost a minimum of $200,000
- Minimal contract with a social agency for Facebook advertising is $80,000
- An SEO content creation agency needs between $60,000 and $220,000
- It has been decided that Adwords cost should be at least 3 times higher than SEO costs
- Customer base = 1300

	TV	SEO	Adwords	Facebook
ROI	0.09	0.14	0.10	0.05
Reach (per $)	2.5	2.1	0.9	3.0

Reflection

- What are **missing** in the above optimisation examples ?
- Will these missing pieces influence the **complexity** of the problems (i.e. make them harder to solve)?
- What will be your **solutions** to address all the challenges above? Hint: simple solutions usually work.
- What are the roles of **data** in optimisation problems?
- What are the challenges or opportunities from big data, *analytics*, and 

for optimisation?

Optimisation Algorithms

- Mathematical programming
  - Linear programming/ Convex Programming
  - Non-linear programming
  - Dynamic programming
  - Branch-and-Bound
- Constraint programming
- Meta-heuristics
  - Simulated annealing
  - Tabu search
  - Evolutionary algorithms
  - Swarm intelligence

Linear vs Non-linear

Unconstrained vs Constrained

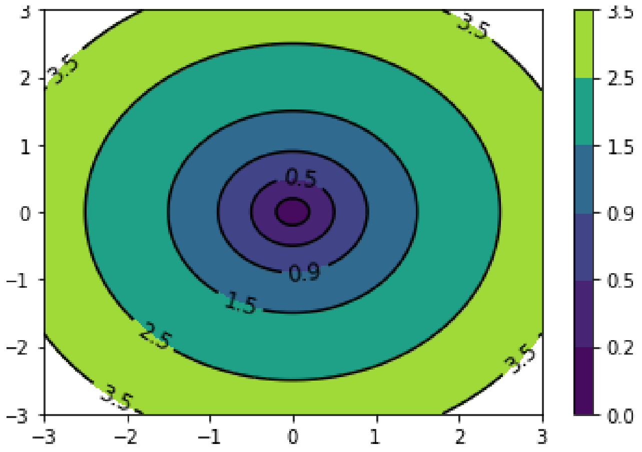
Continuous vs Discrete

Deterministic vs Stochastic

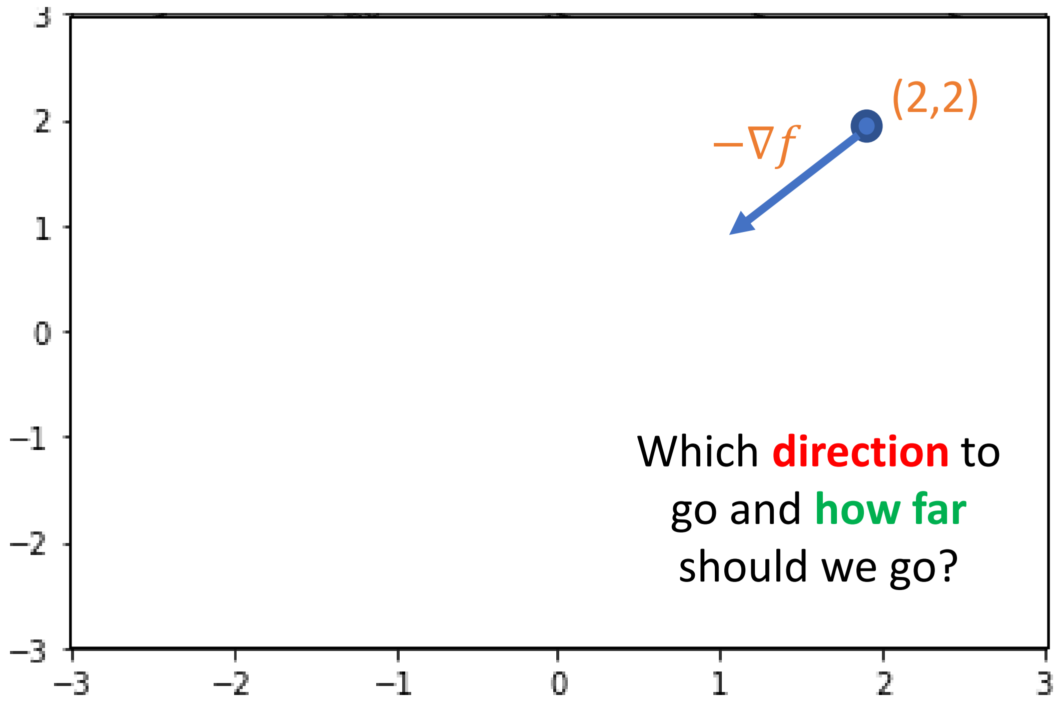
Single Objective vs Multiple Objective

How Optimisation Algorithms Work? (1)

$$minimise\ f(x,y) = x^2 + y^2$$
$$Subject\ to:$$
$$0 \le x, y \le 3$$

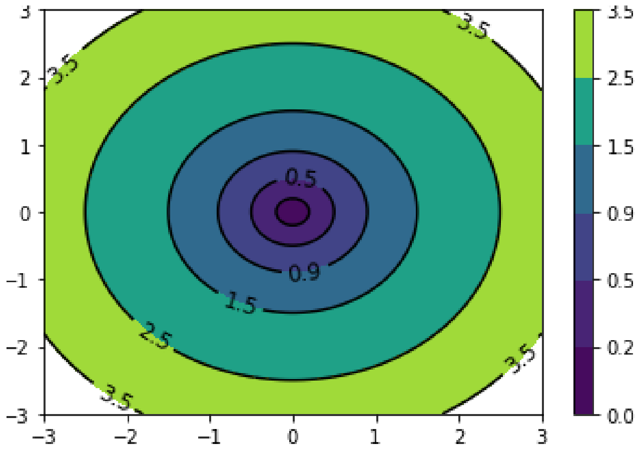


Gradient-based Optimisation

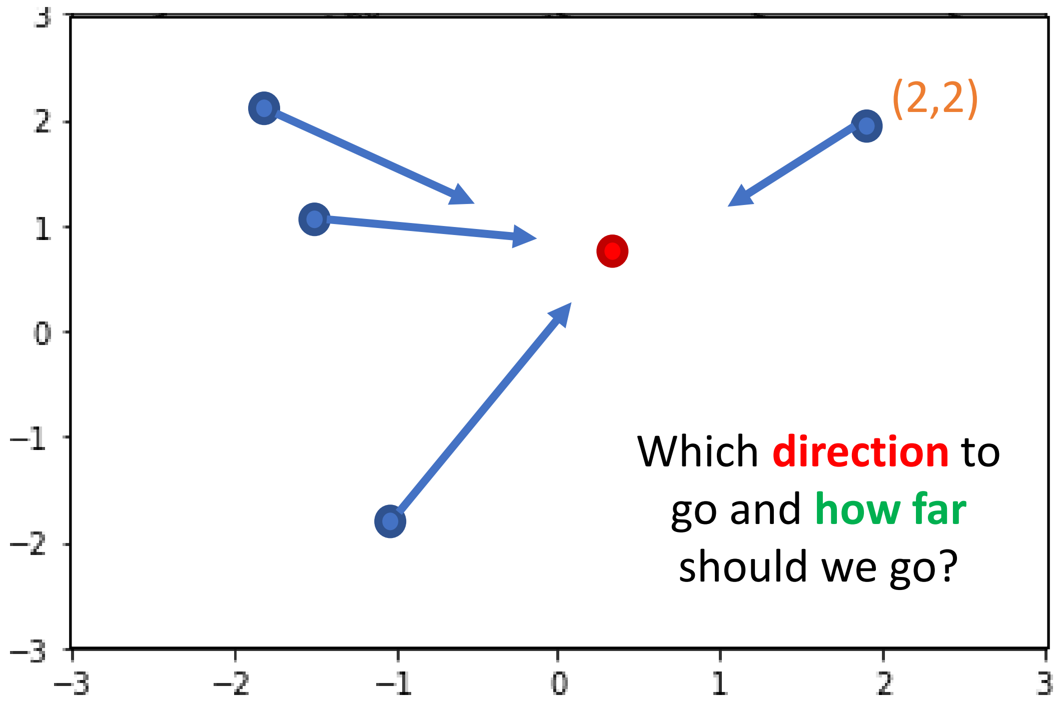


How Optimisation Algorithms Work? (2)

$$minimise\ f(x, y) = x^2 + y^2$$
$$Subject\ to:$$
$$0 \le x, y \le 3$$

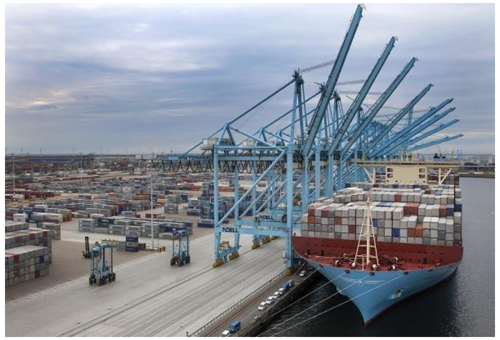


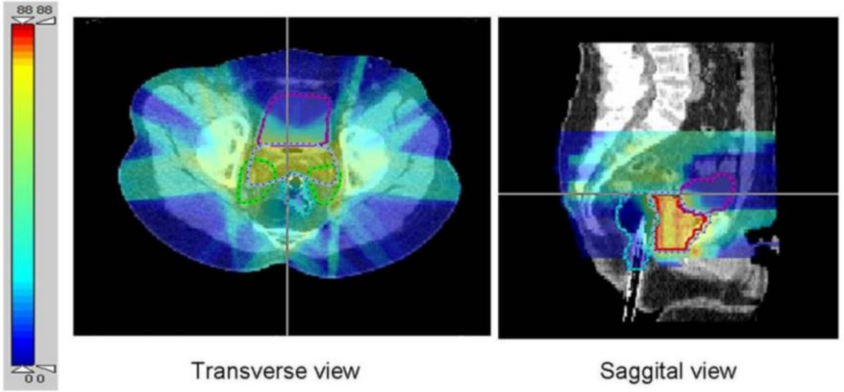
Stochastic Optimisation (e.g. PSO)



Challenges (1)

- Discrete decision variables:
  - Whether to build a warehouse – Supply chain network design
  - Determine the sequence to pick up and delivery – fleet management
  - Quay crane scheduling
  - Portfolio optimisation
  - Operating room surgical scheduling
  - Cancer treatment optimisation





LA TROBE UNIVERSITY

All kinds of clever

Challenges (2)

- Uncertainty
  - Stochastic demand in aggregate production planning
  - Machine breakdowns
  - Emergency evacuation
  - Traffic conditions
  - Employee sickness
  - Customer behaviour
  - Risk management
  - Etc.

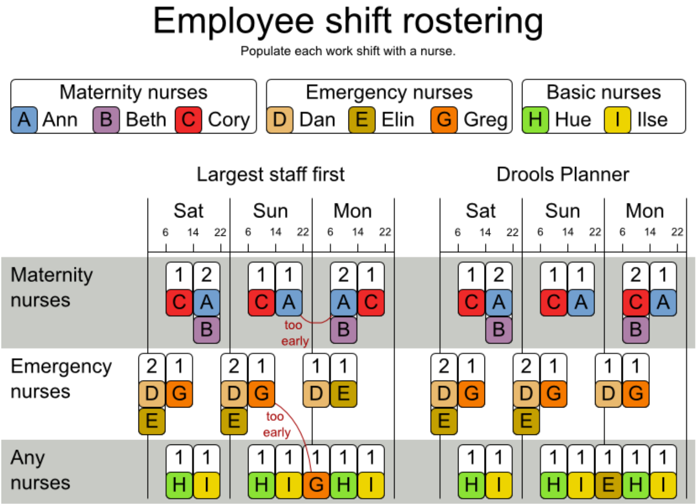
Challenges (3)

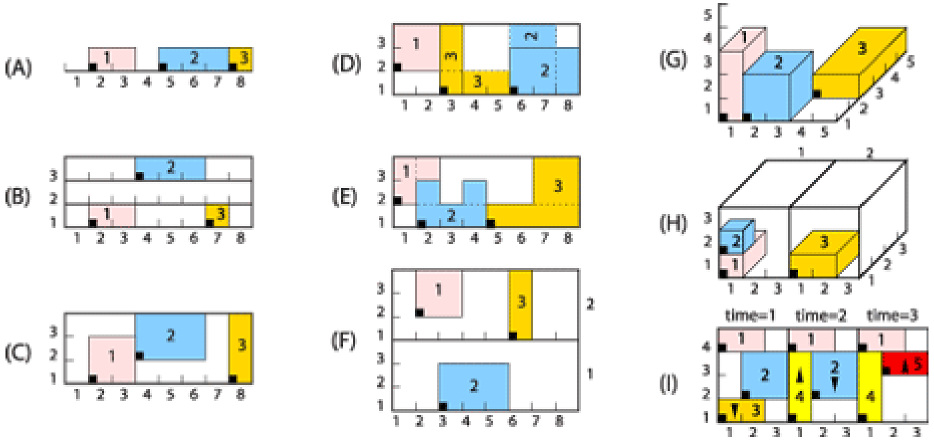
- Highly constrained → Constraint programming (many good solvers are available):
  - Time-tabling problems
  - Vehicle routing problems
  - Pattern mining/ Constrained clustering
  - Nurse rostering

Google OR-Tools

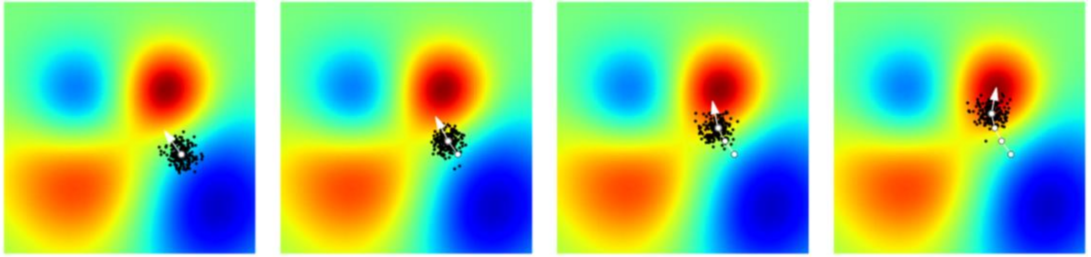
MiniZinc

Gecode



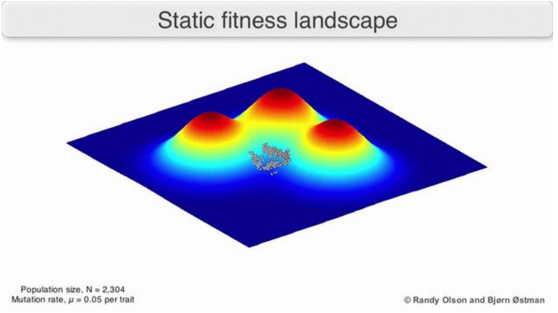


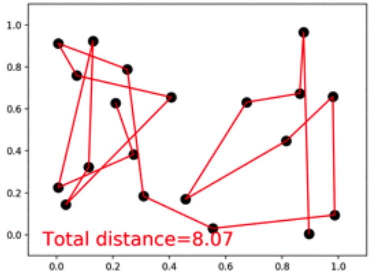
Meta-heuristics



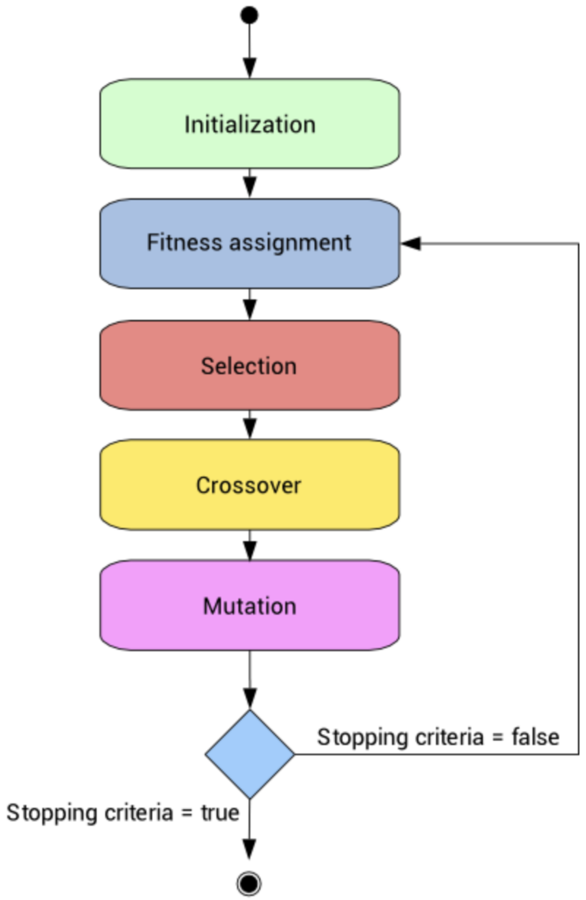
https://blog.openai.com/evolution-strategies/

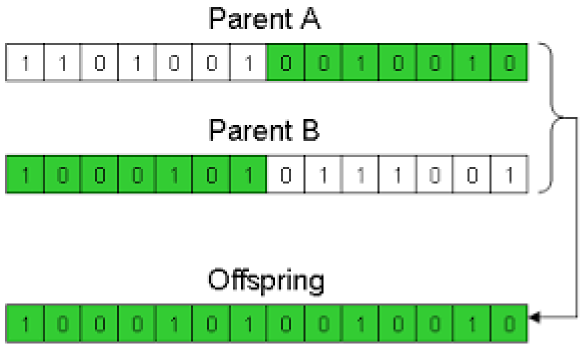
- Some problems are just too hard to solve to optimality – the computational time is exponentially grown as the size of the problem increases
  - Discrete optimisation/ Combinatorial optimisation
  - Complex, non-linear objective functions
- Meta-heuristics can find “good” solutions in reasonable running times
  - Genetic algorithms: provide decent results in many applications
  - Particle swarm optimisation: works well with continuous variables
  - Natural evolutionary strategy: provide robust results and can be easily paralellised
  - Ant colony optimisation: work best for network problems
  - Software: ECJ (Java), DEAP (Python), HeuristicLab (C#), NeverGrad (Facebook)



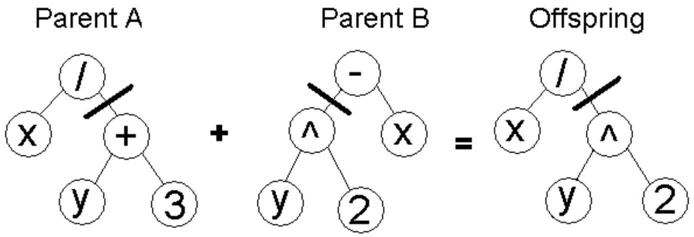


Evolutionary Algorithms (GA, GP, ES, etc.)

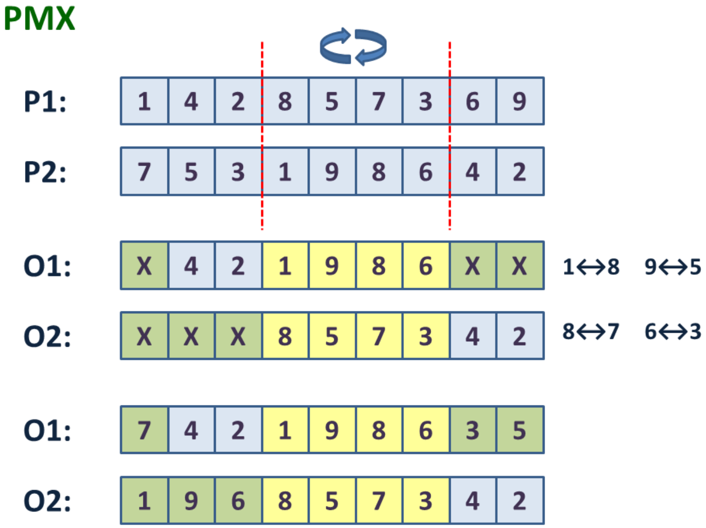




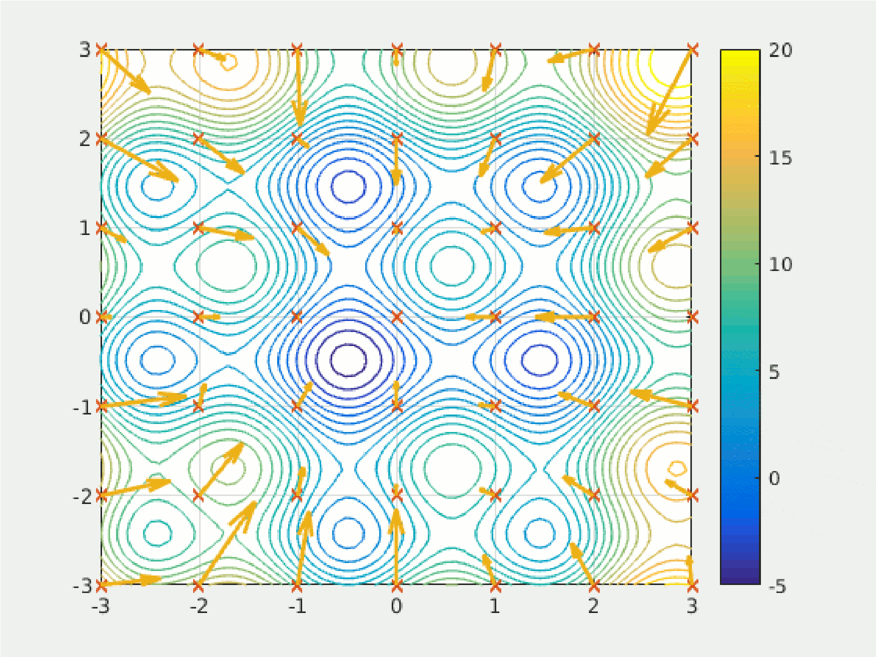
One-point crossover



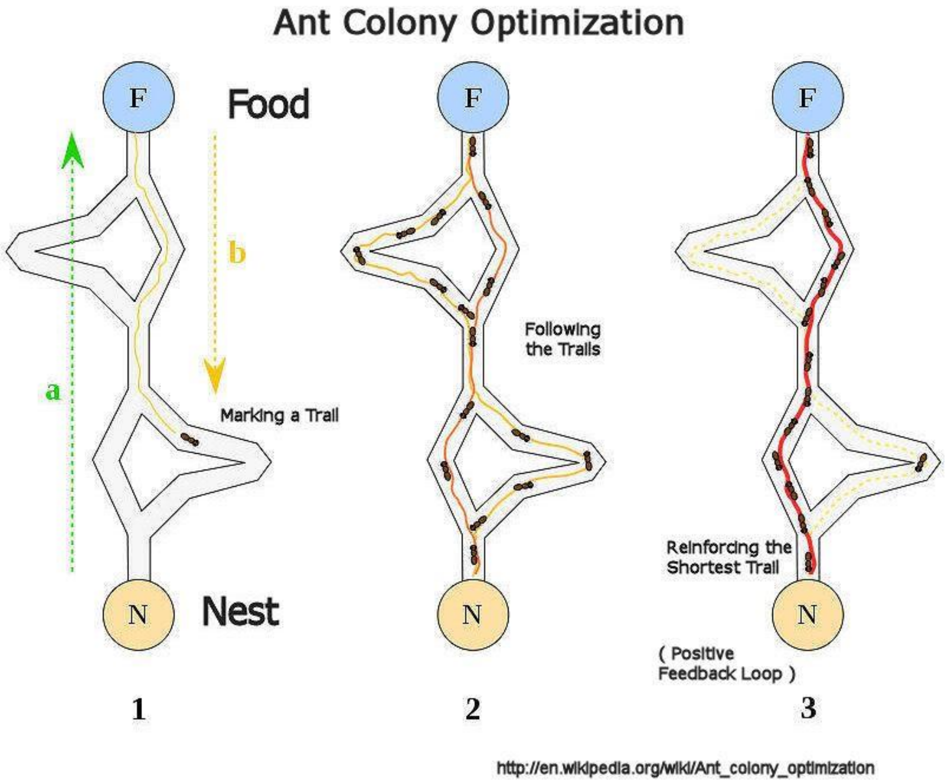
Sub-tree crossover



Swarm Intelligence



Particle Swarm Optimisation
(individual experience vs swarm experience)



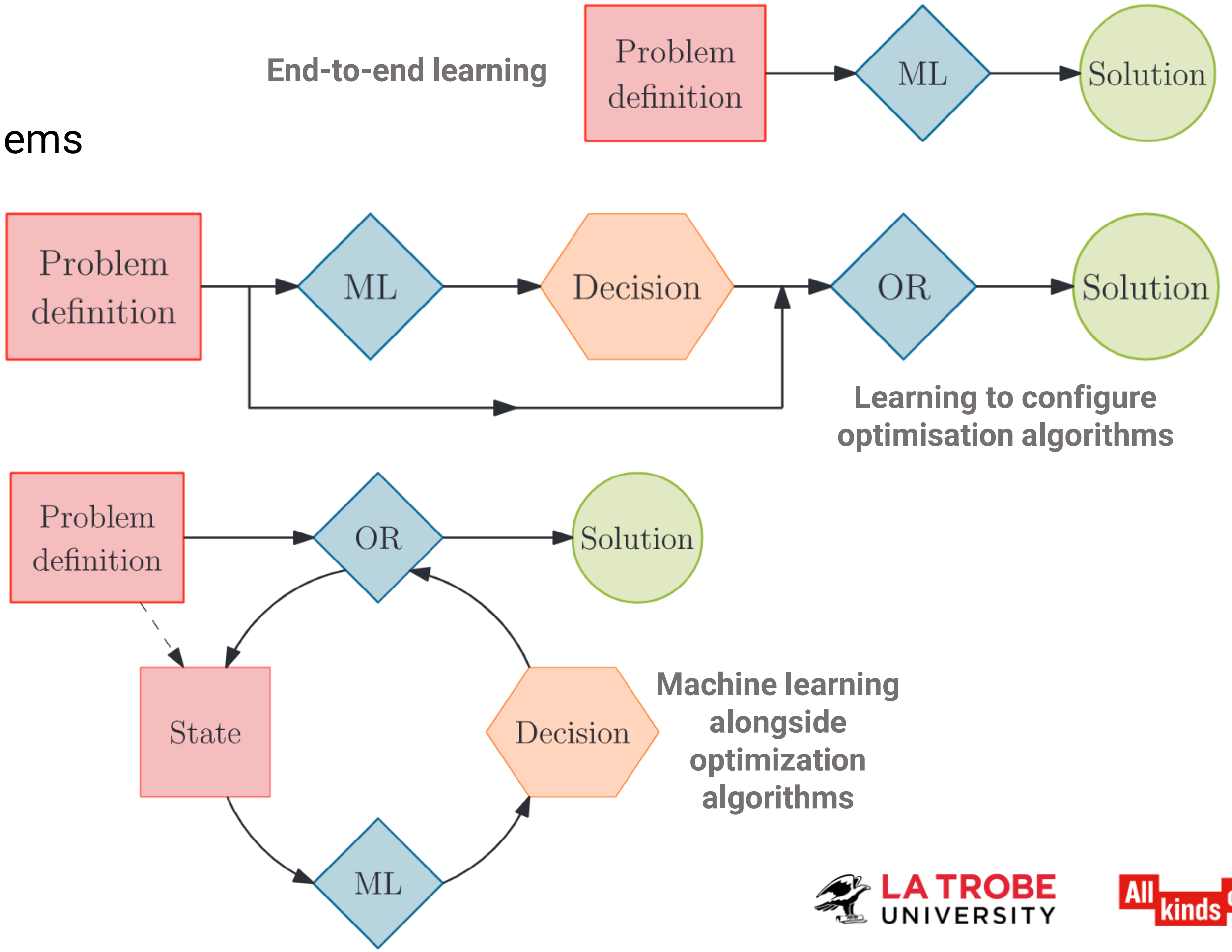
Heuristic Journey

- **Heuristics** refer to experience-based techniques to find good solutions for computational problems.
- **Meta-heuristics** are optimisation methods designed to deal with hard optimisation problems. Different from heuristics, meta-heuristics are more abstract and usually make no or very few assumptions about the problems to be solved.
- **Hyper-heuristics (HH)** are a relative new research area which focuses on exploring the "heuristic search space" of the problems instead of the solution search space in the cases with heuristics and meta-heuristics.

Machine Learning for Combinatorial Optimisation (MLCO)

- Use machine learning algorithms to solve CO problems
- Existing studies:
  - Supervised learning
  - Reinforcement learning
- Applications:
  - TSP, Vehicle routing problems
  - Job shop scheduling

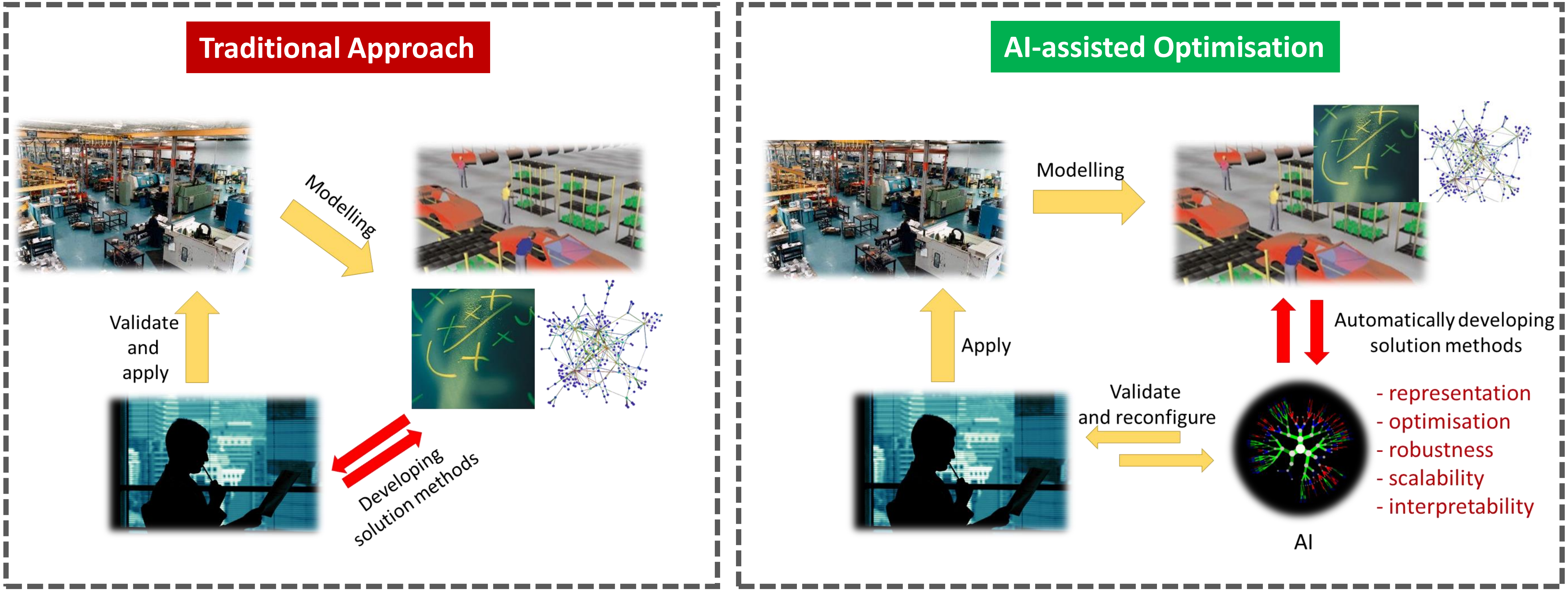
Bengio, Y et al., "Machine learning for combinatorial optimization: A methodological tour d'horizon", EJOR, 20.



LA TROBE UNIVERSITY

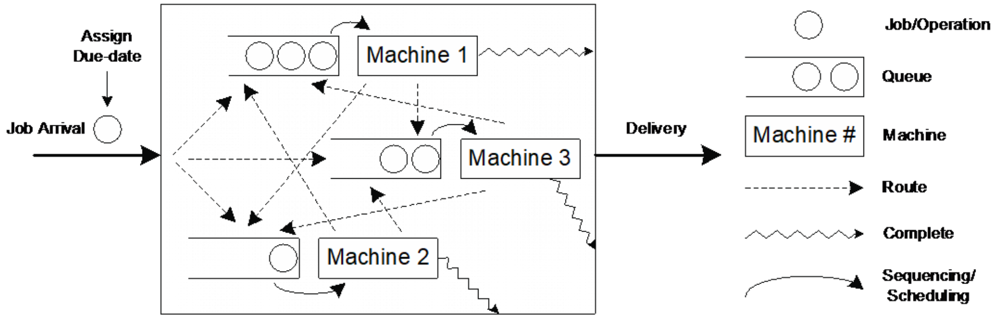
All kinds of clever

Our Approach

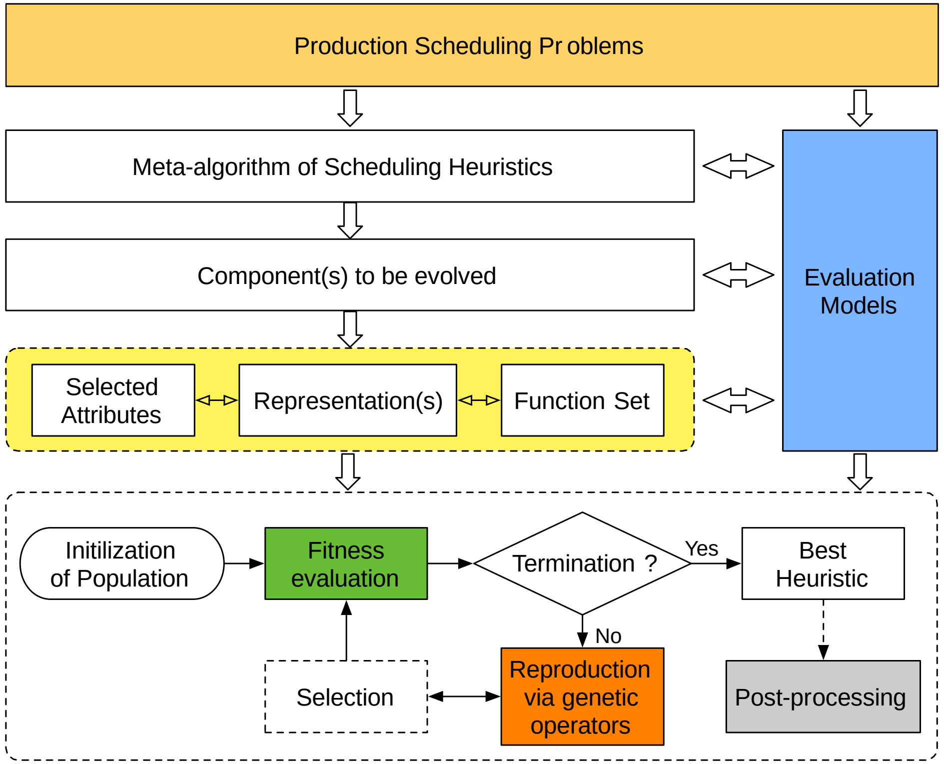


Case study: Dynamic Production Scheduling

- Scheduling is an essential task in production planning and control
  - Optimise resource utilization
  - Ensure delivery performance
  - Coordinate with other logistical activities
- Multiple jobs to be allocated to multiple machines
- Precedence-constrained must be satisfied
- Multiple conflicting objectives
- NP-hard problem



Evolutionary Learning Framework for Learning Scheduling Heuristics



Nguyen et al., "Genetic programming for production scheduling: a survey with a unified framework", Complex and Intelligent Systems, 2017.

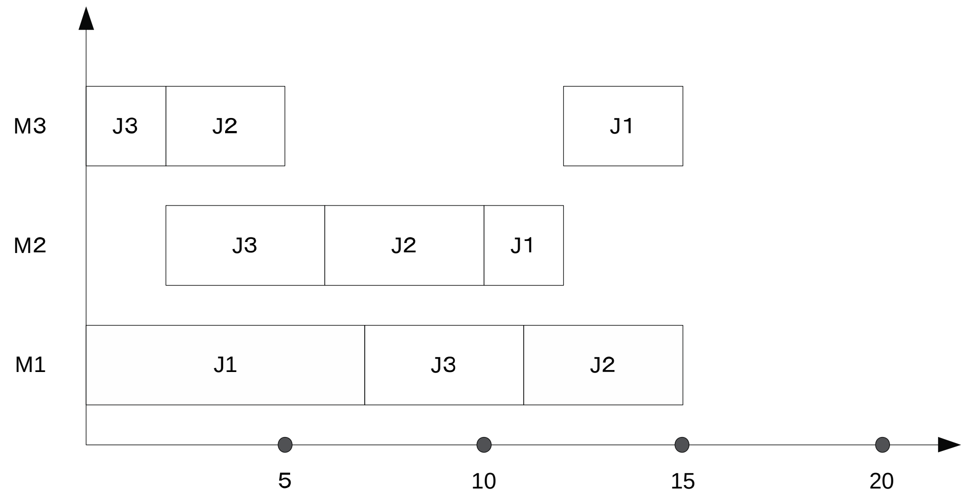
Static Job Shop Scheduling

- Discover construction heuristics
- Discover improvement heuristics
- Augment optimisation algorithm

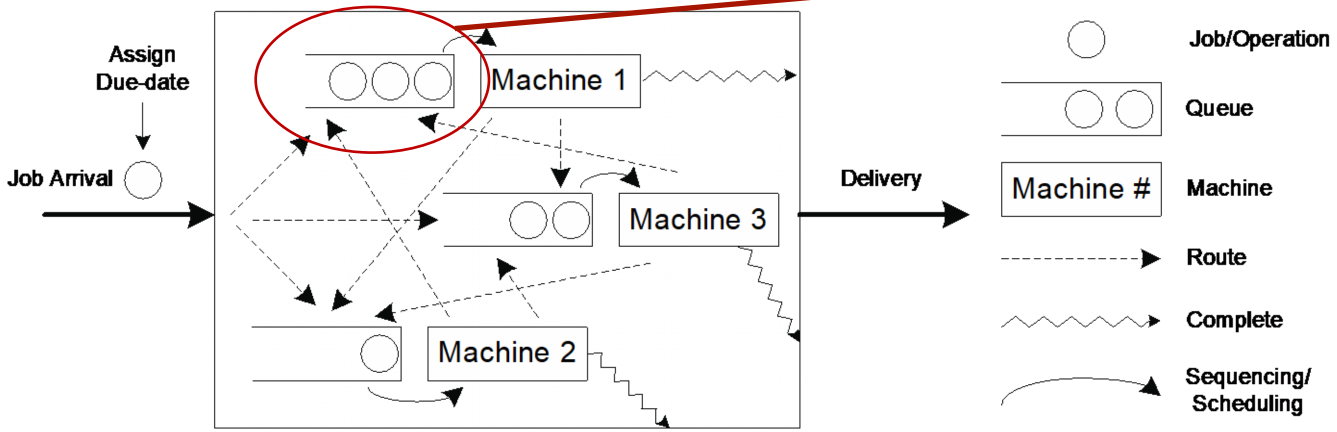
EXAMPLE OF A STATIC JSS PROBLEM INSTANCE ($N = 3, M = 3$)

j	$route_j$	r_j	$time_j$	w_j	d_j	C_j	T_j
#1	1,2,3	0	7,2,3	4	13	15	2
#2	3,2,1	0	3,4,4	2	10	15	5
#3	3,2,1	0	2,4,4	1	12	11	0

$C_{max} = 15$, $T_{max} = 5$, $TWT = 18$



Discover Construction Heuristics



Dispatching rule decides quality of constructed schedule.

Existing rules are simple functions to assign priorities to jobs/operations

Minimum Slack	$\max(d_j - p_j - t, 0)$
Weighted Shortest Processing Time	$\frac{w_j}{p_j}$
Apparent Tardiness Cost	$\frac{w_j}{p_j} \exp\left(-\frac{\max(d_j - p_j - t, 0)}{K\bar{p}}\right)$

Is it possible to design a better priority function?

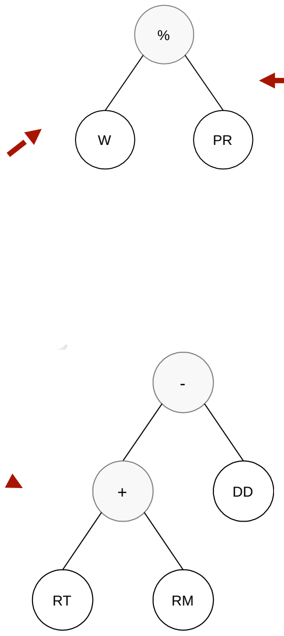
Representation in EL

Terminal set

Notation	Description
RJ	operation ready time
RO	number of remaining operations of job
RT	work remaining of job
PR	operation processing time
W	weight
DD	due date
RM	machine ready time
#	constant Uniform [0,1]

Function set

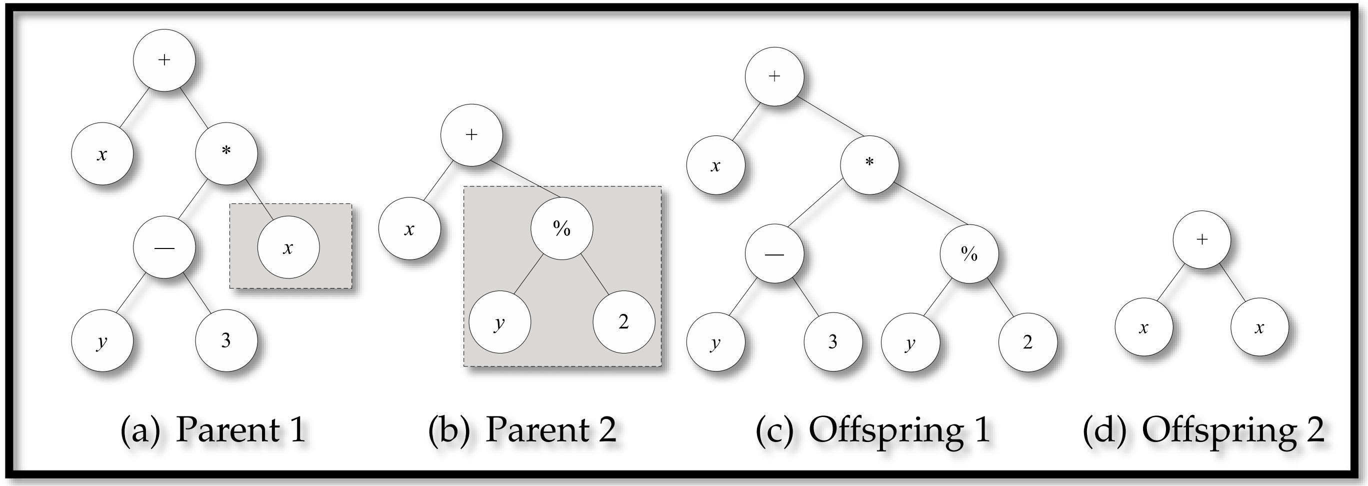
+, -, %, *, min, max, etc.



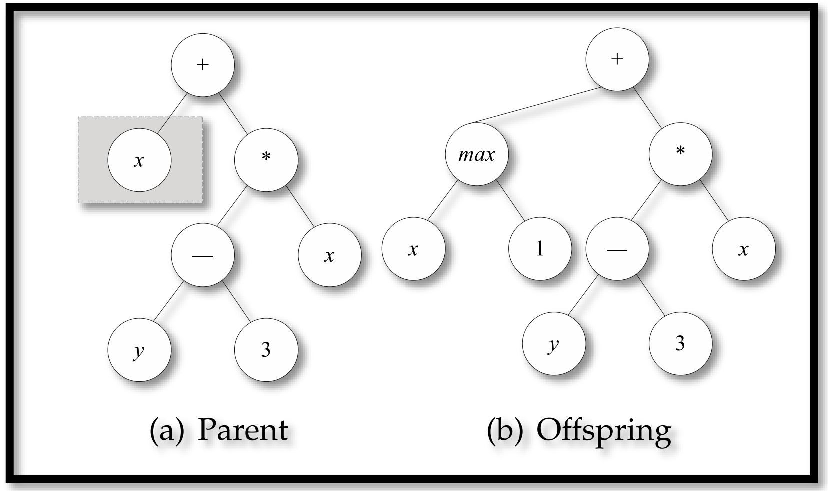
$$\frac{w_j}{p_j}$$

Genetic Operators

Sub-tree Crossover



Sub-tree Mutation

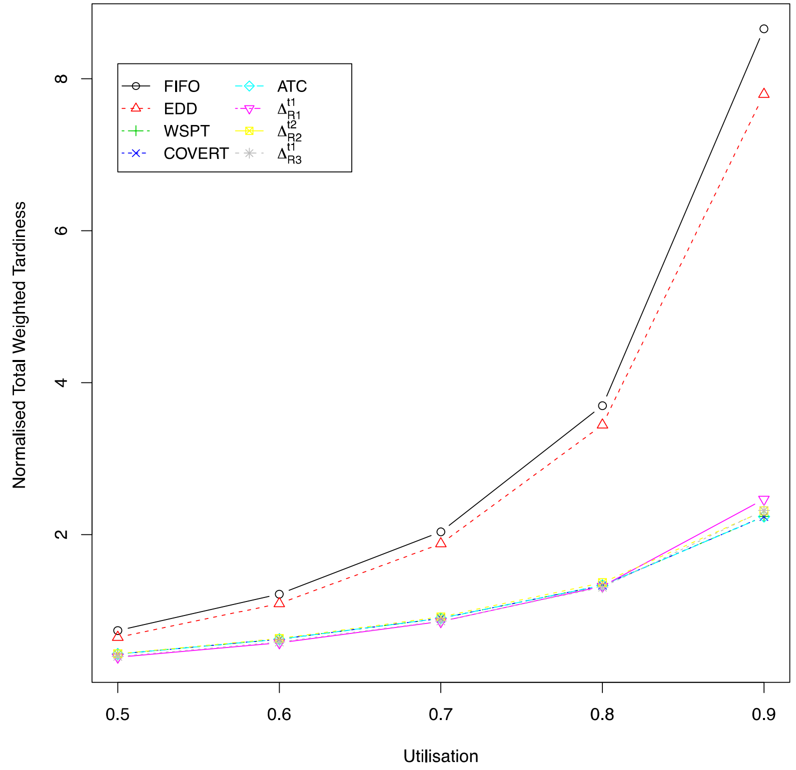


Compared to Existing Rules

Evolved Heuristics

instance	optimal	$\Delta^{t1}_{R_1}$	$\Delta^{t2}_{R_2}$	$\Delta^{t1}_{R_3}$	ATC$^{\alpha}$	COVERT$^{\alpha}$	GA-1	GA-2	LSRW(15)	LSRW(200)
la16	1170	$2000^{(a)}$	2521	2477	2360	2360	1623	1619	*	*
la17	900	$1623^{(c)}$	2107	$1555^{(ac)}$	1936	1777	1464	1779	977	*
la18	929	1764	1662	$1325^{(a)}$	1439	1694	1248	1248	946	*
la19	948	1884	2284	$1505^{(bc)}$	$1439^{(ab)}$	2189	1696	1671	966	951
la20	809	2702	2898	$1784^{(a)}$	1866	2069	1402	1402	819	*
la21'	464	$1280^{(c)}$	2293	$1047^{(ac)}$	1530	1530	1044	1286	*	*
la22'	1068	1931	2270	2600	2960	$1870^{(a)}$	1550	1909	1149	1086
la23'	837	$2053^{(a)}$	2100	2131	2605	2605	1094	1094	875	875
la24'	835	1948	2692	$1136^{(abc)}$	1875	1819	1403	1479	844	*
orb01	2568	5247	5680	$4210^{(ac)}$	6289	4583	4005	4865	2726	2616
orb02	1412	2526	$2461^{(a)}$	2797	3022	2908	2143	2075	1434	1434
orb03	2113	4276	3968	$2880^{(ac)}$	4215	4215	2866	4647	2289	2204
orb04	1623	3285	3060	$2280^{(abc)}$	3945	3945	2326	3188	1816	1674
orb05	1593	3360	4031	$1986^{(abc)}$	3511	3573	2533	2673	1802	1662
orb06	1792	3759	4098	$3371^{(c)}$	$3146^{(a)}$	$3146^{(a)}$	3047	3689	1852	1802
orb07	590	1169	$834^{(abc)}$	$922^{(bc)}$	1324	1239	927	982	619	618
orb08	2429	$2554^{(abcd)}$	5147	4570	$3777^{(bc)}$	4404	3792	4103	2717	2554
orb09	1316	$2388^{(abc)}$	2949	$2415^{(c)}$	2679	3190	2061	2628	1449	1334
orb010	1679	3075	3186	$2802^{(ac)}$	3418	3109	2217	2913	1837	1775

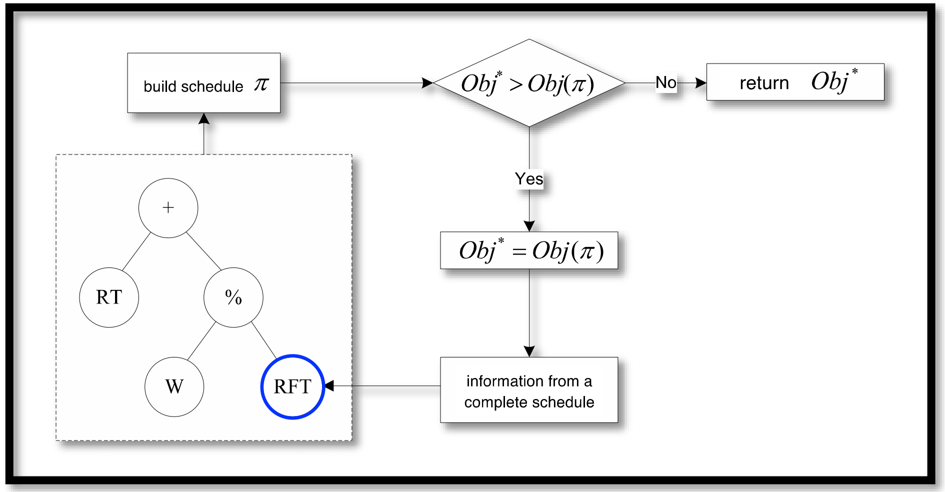
"GA-1" and "GA-2" represent GA-(R&M/COVERT) and GA-(R&R) [20]; ATC$^{\alpha}$ and COVERT$^{\alpha}$ showed the results from ATC and COVERT with $\alpha = 0.4$.
"*" means the method found the optimal solution; "a" means that it is the best rule among the three evolved rules for the instance;
"b" means that the evolved rule is better than GA-(R&M/COVERT) for the instance; "c" means that the evolved rule is better than GA-(R&R) for the instance;
"d" means that the evolved rule is better than LSRW(15) for the instance; and "e" means that the evolved rule is better than LSRW(200) for the instance



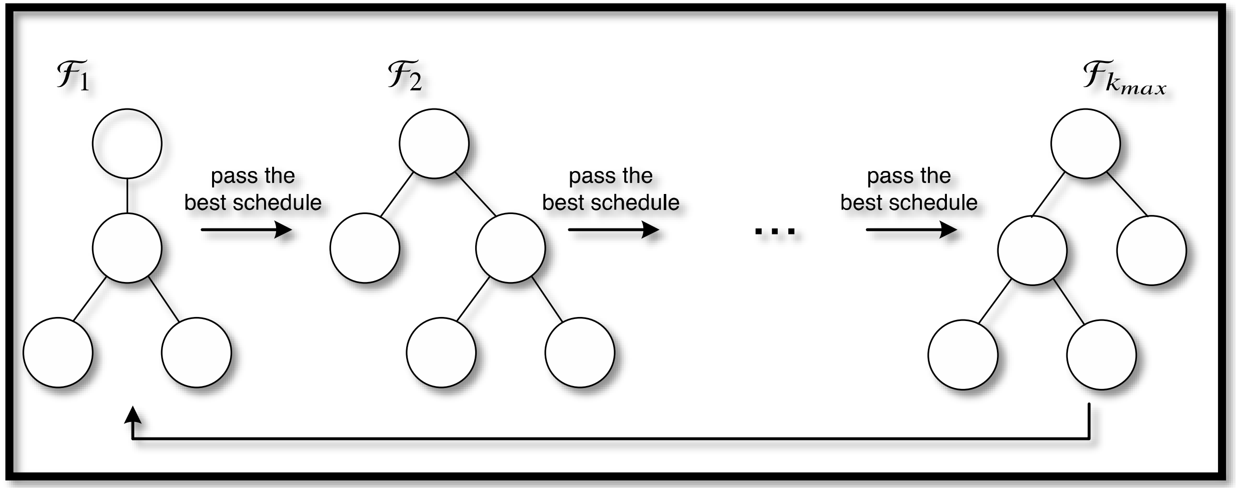
Nguyen et al., "A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem", IEEE Trans. Evolutionary Computation, 2013.

Discover Improvement Heuristics

Using Feedback from Existing Schedules

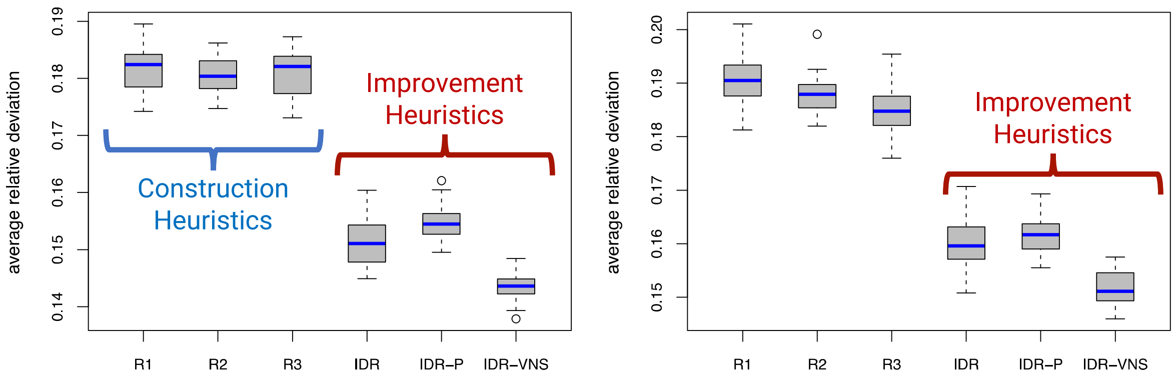


Multiple Ways to Construct/Improve Schedules

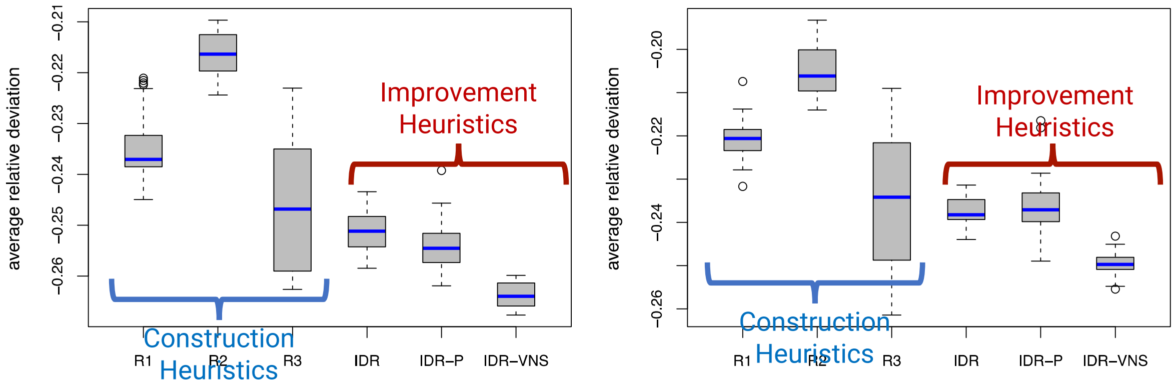


Similar representation, different meta-algorithms

Experiments



(a) Training set/ $Jm||C_{max}$ (b) Test set/ $Jm||C_{max}$



(c) Training set/ $Jm||\sum w_jT_j$ (d) Test set/ $Jm||\sum w_jT_j$

- Evolved improvement heuristics outperform construction heuristics in most cases
- Execution times are not significantly increased
- Requires some major changes in the meta-algorithms
- Still no way to prove optimality

Nguyen et al., "Learning iterative dispatching rules for job shop scheduling with genetic programming", The International Journal of Advanced Manufacturing Technology, 2013.

Augment Optimisation Algorithms

Problem formulation

$$\text{minimize } C_{\max}$$

subject to

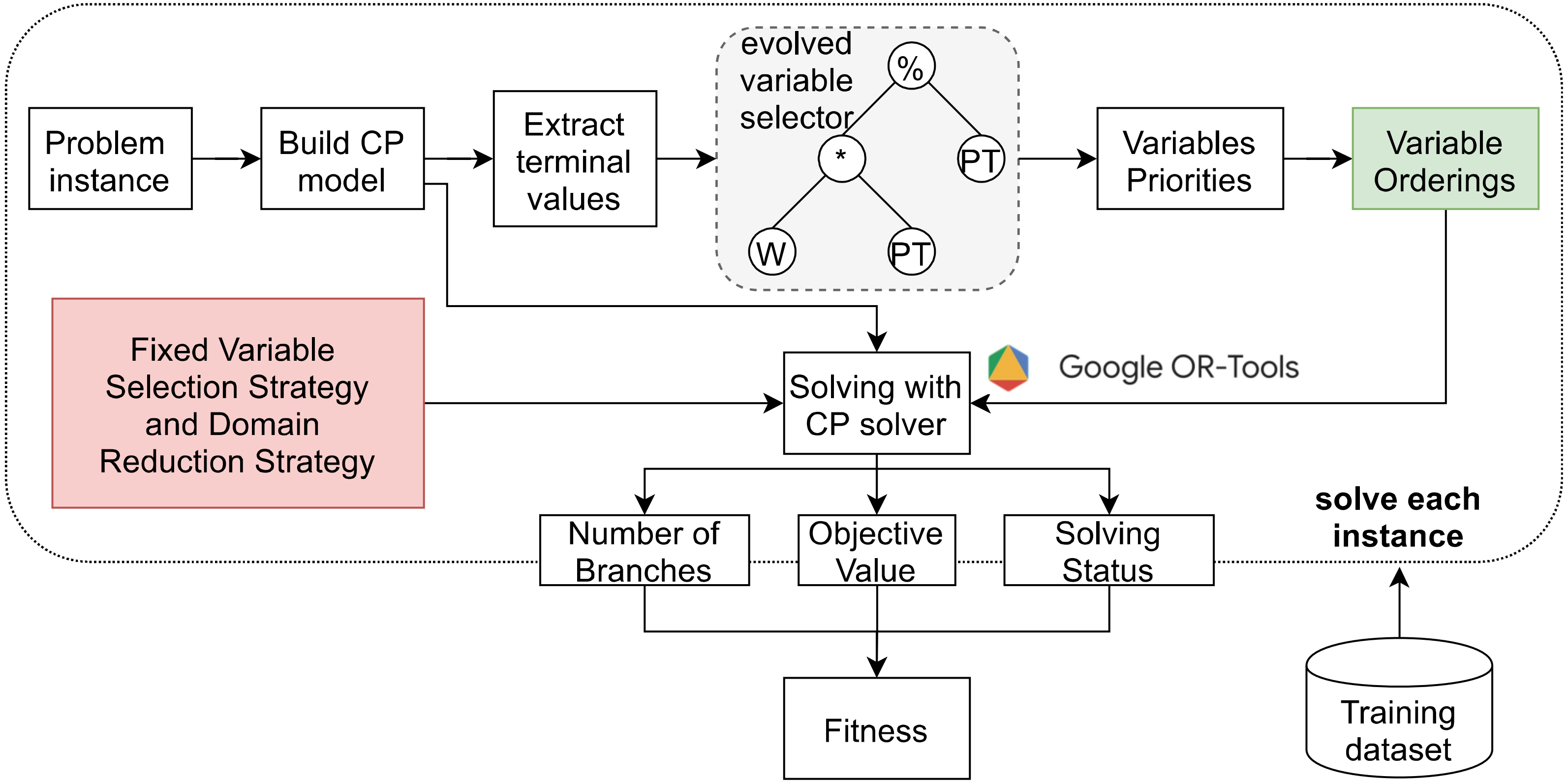
$$
\begin{aligned}
y_{kj} - y_{ij} &\geq p_{ij} && && \text{for all } (i,j) \to (k,j) \in A \\
C_{\max} - y_{ij} &\geq p_{ij} && && \text{for all } (i,j) \in N \\
y_{ij} - y_{il} &\geq p_{il} && \text{or} \quad y_{il} - y_{ij} \geq p_{ij} && \text{for all } (i,l) \text{ and } (i,j), \quad i = 1, \ldots, m \\
y_{ij} &\geq 0 && && \text{for all } (i,j) \in N
\end{aligned}
$$

Python, Minizinc, OPL, etc.

Solver

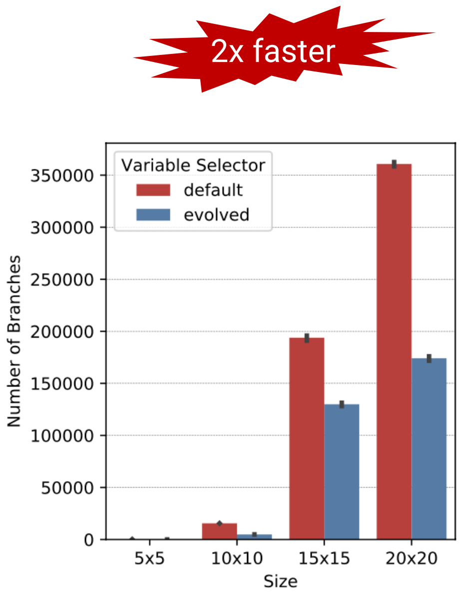
- Can only solve small instances
- Take a long time to produce "*acceptable*" solutions for medium/large instances

Genetic-based Constraint Programming (GCP)

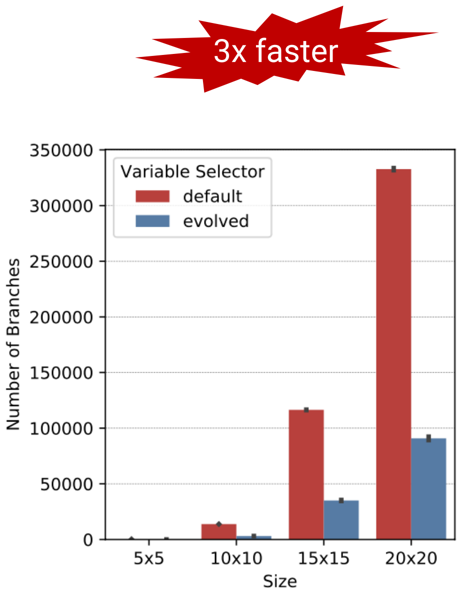


- CP search algorithm is the meta-algorithm
- If stopped after the first feasible solution is founded, it is similar to construction heuristics
- If solutions are improved by CP, it is similar to improvement heuristics
- Possible to prove optimality for some instances

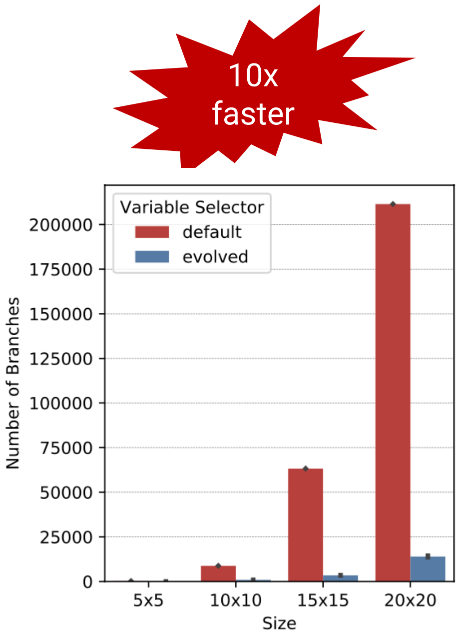
Experiments (1)



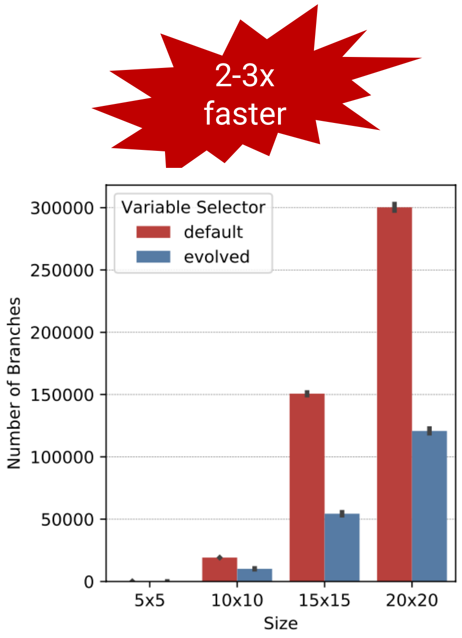
(a) Cmax



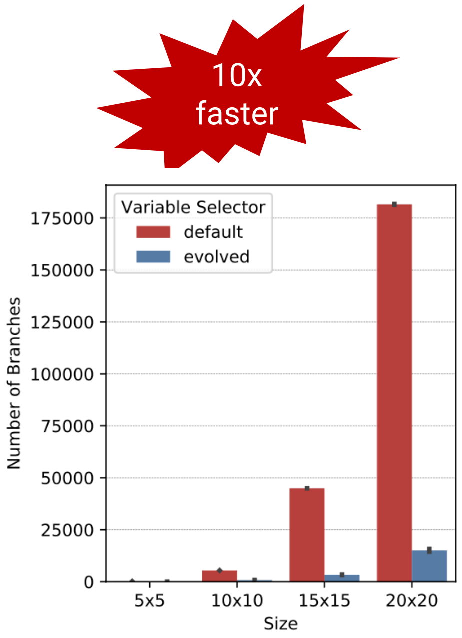
(b) Tmax, $h = 1.5$



(c) Tmax, $h = 1.6$



(d) TWT, $h = 1.5$

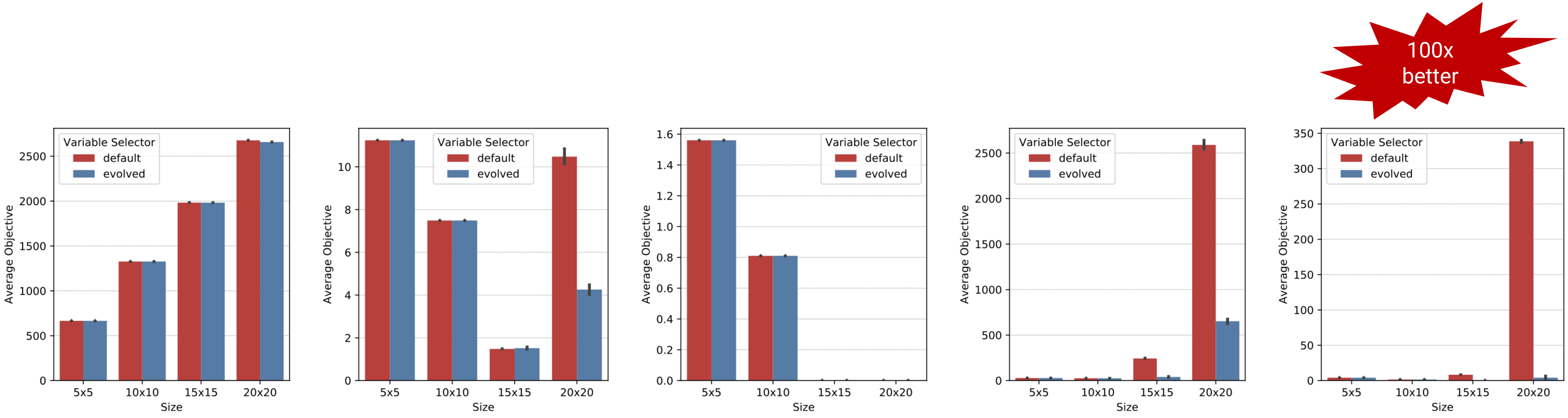


(e) TWT, $h = 1.6$

Computational Efforts (# of branches)

Nguyen et al., "A genetic programming approach for evolving variable selectors in constraint programming", IEEE Trans. Evolutionary Computation, 2021.

Experiments (2)

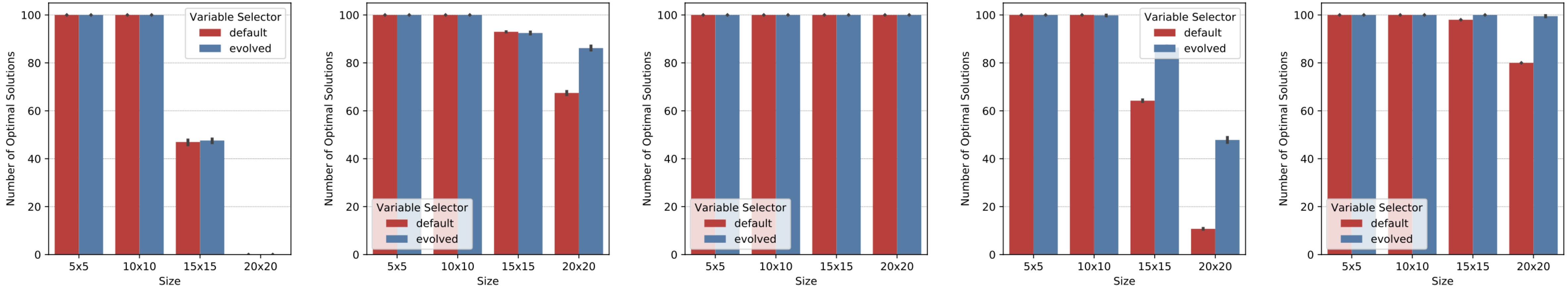


(a) Cmax (b) Tmax, $h = 1.5$ (c) Tmax, $h = 1.6$ (d) TWT, $h = 1.5$ (e) TWT, $h = 1.6$

Solution Quality (Objective Values)

Nguyen et al., “A genetic programming approach for evolving variable selectors in constraint programming”, IEEE Trans. Evolutionary Computation, 2021.

Experiments (3)



(a) Cmax (b) Tmax, $h = 1.5$ (c) Tmax, $h = 1.6$ (d) TWT, $h = 1.5$ (e) TWT, $h = 1.6$

Optimality (# proven optimal solutions)

Nguyen et al., "A genetic programming approach for evolving variable selectors in constraint programming", IEEE Trans. Evolutionary Computation, 2021.

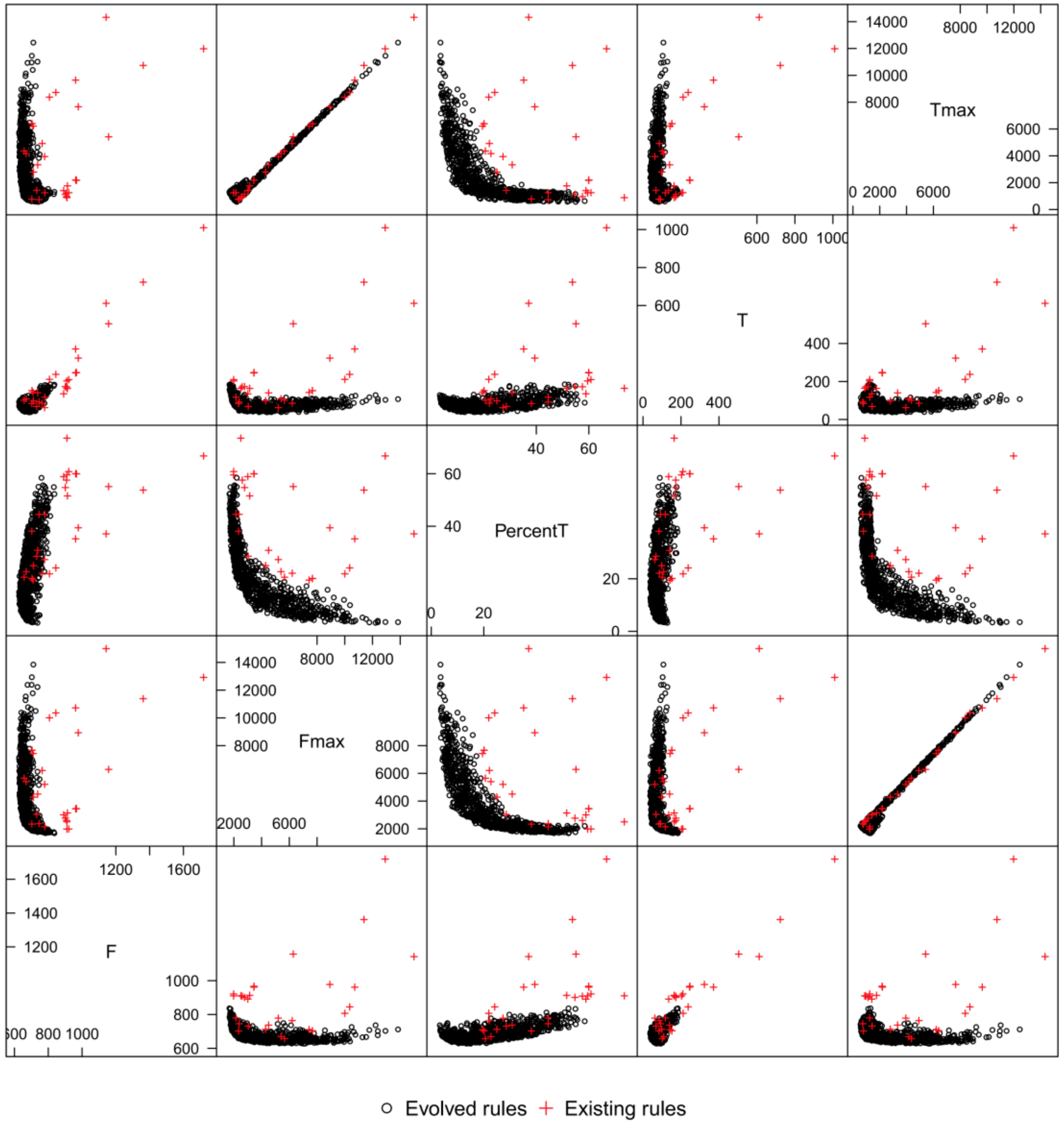
Dynamic Job Shop Scheduling

- Dynamic and random job arrivals
- Stochastic processing times
- Machine breakdowns
- Material availability

Previous representation and priority rules are perfect for these scenarios

- **Reactive scheduling**
- **Interpretable**
- **Efficient**

Multiple Conflicting Objectives

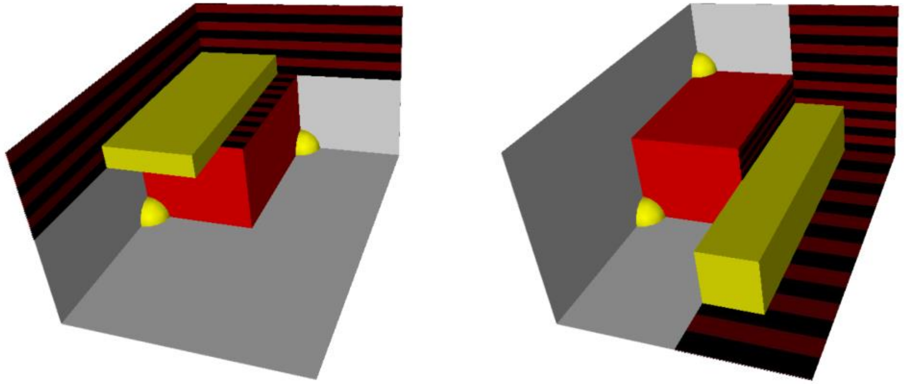


Nguyen et al., "Dynamic Multi-objective Job Shop Scheduling: A Genetic Programming Approach", Automated Scheduling and Planning, 2013.

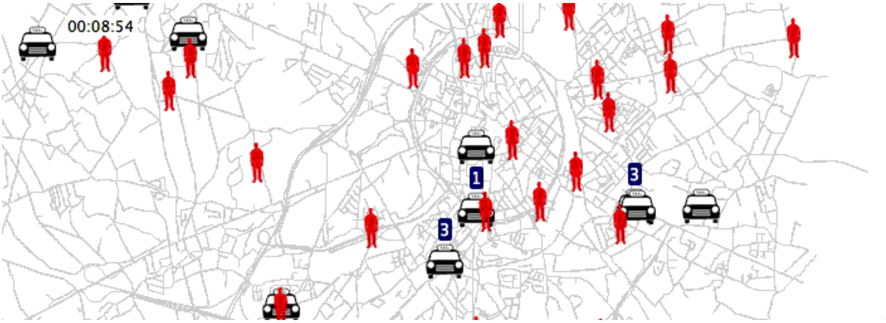
LA TROBE UNIVERSITY

All kinds of clever

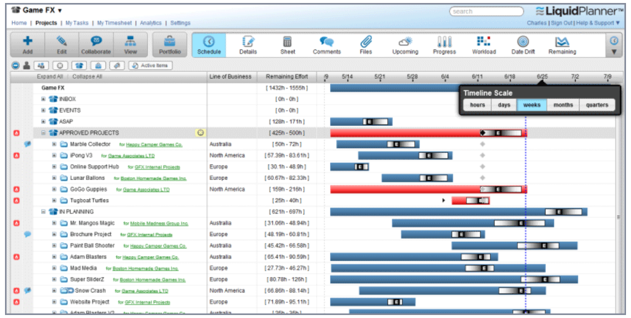
Other Applications



Matthew Hyde (2010) "A genetic programming hyperheuristic approach to automated packing". PhD thesis, University of Nottingham.



Rinde R. S. van Lon, Juergen Branke, Tom Holvoet (2017) "Optimizing agents with genetic programming: an evaluation of hyper-heuristics in dynamic real-time logistics", Genetic Programming and Evolvable Machines

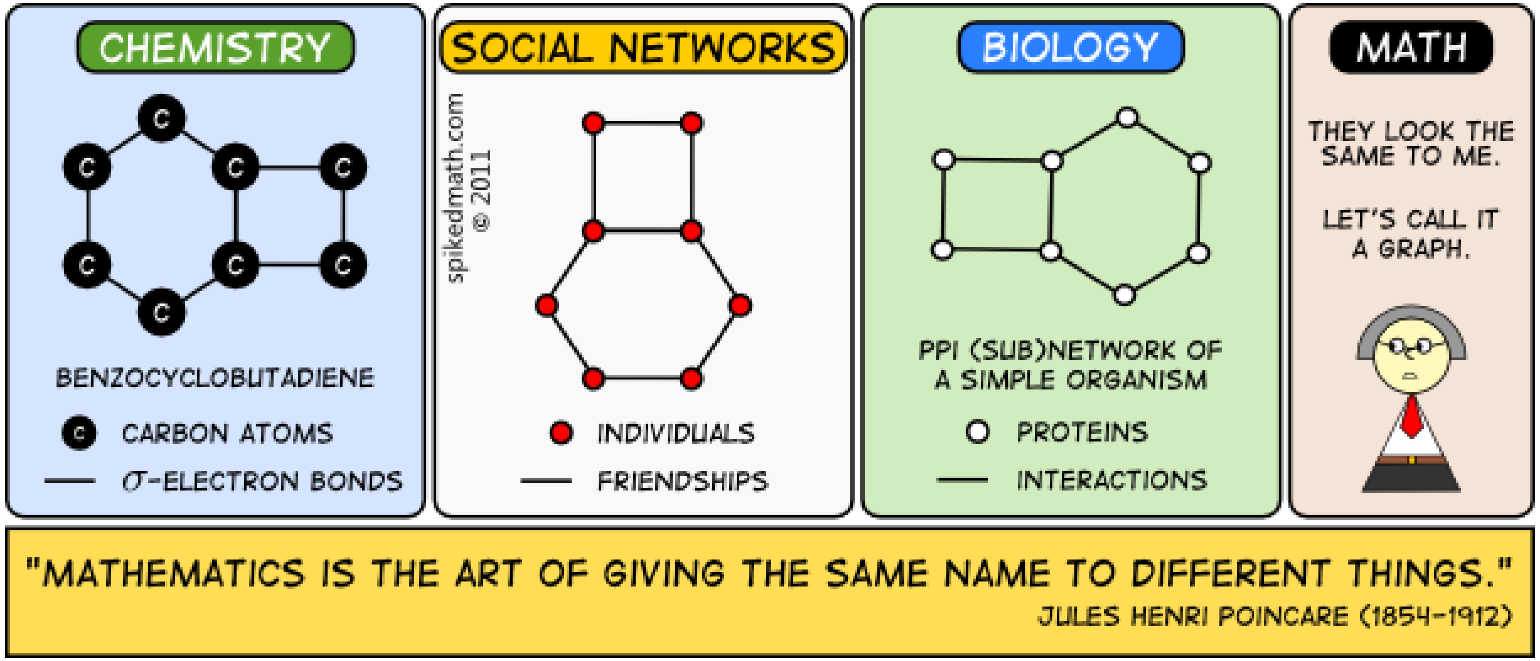


Shelvin Chand, Quang Huynh, Hemant Singh, Tapabrata Ray, Markus Wagner (2018) "On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems". Information Science.

Su Nguyen, Dhananjay Thiruvady, Andreas Ernst, Damminda Alahakoon (2018) "Genetic programming approach to learning multi-pass heuristics for resource constrained job scheduling", GECCO'18

Future Research Directions

- Can we use AI to support/replace experts to design an efficient way to solve real-world optimisation problem?
- Can we transfer the knowledge what we have learned from solving an optimisation problems to different but related optimisation problems?
- Can we learn to solve multiple optimisation tasks at the same time?
- Can we freely formulate the optimisation problems and freely allow AI/ML to discover how to solve them?
- Can AI/ML learn how to present its solutions to us in a friendly and interactive way?
- Can human and AI systems communicate and co-design a solution for complex optimisation task?



https://caseine.org/course/view.php?id=42&section=7

THANKS

My Email: p.nguyen4@latrobe.edu.au