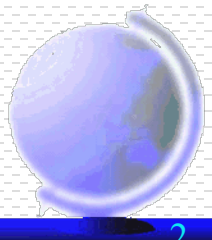
《经典运筹学问题与模型》课程

第二章 最优化问题与算法概述

唐加福 朱晗
东北财经大学管理科学与工程学院
jftang@mail.neu.edu.cn hanzhu@dufe.edu.cn
Tel:0411-84711310 / 84713592

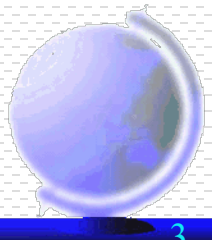
第 2 章 最优化问题与算法概述

- 什么是最优化问题—从现实问题入手
- 优化问题分类
- 约束优化与无约束优化
- 组合最优化问题
- 系统建模与数学模型
- 优化算法与分类



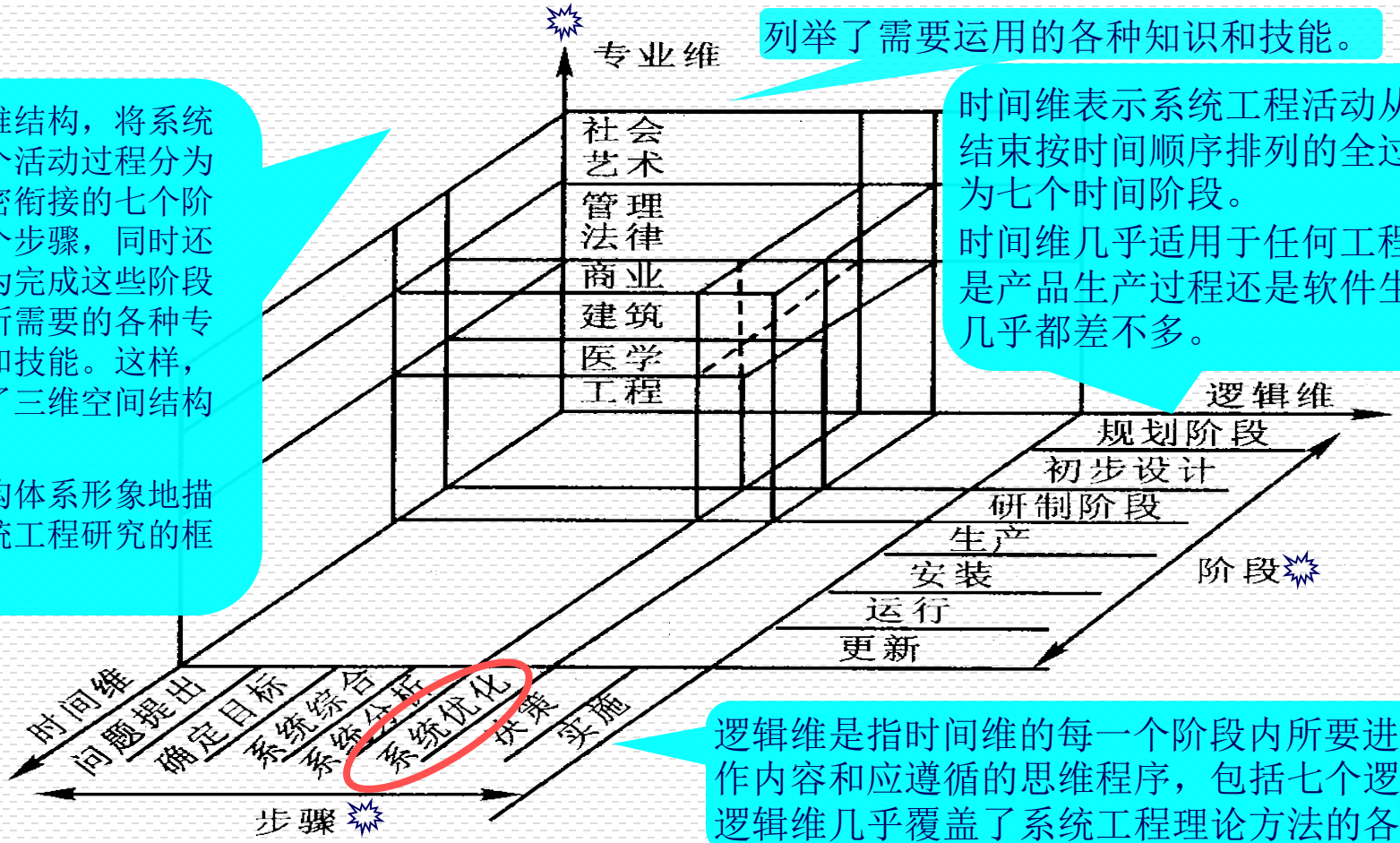
课程导论-目的

- 理解优化问题的基本思维方式
- 了解各种典型的优化问题及对应现实问题
- 了解典型优化问题的特征、模型及其求解方法
- 提高分析问题、解决问题的能力
- 为今后的应用与研究奠定基础



最优化的重要性-从系统工程方法论说起

- **霍尔三维构结 (Hall three dimensions structure)**
  - 是美国系统工程专家霍尔(A・D・Hall)于1969年提出的一种系统工程方法论。它的出现，为解决大型复杂系统的规划、组织、管理问题提供了一种统一的思想方法，因而在世界各国得到了广泛应用。



列举了需要运用的各种知识和技能。

霍尔三维结构，将系统工程整个活动过程分为前后紧密衔接的七个阶段和七个步骤，同时还考虑了为完成这些阶段和步骤所需要的各种专业知识和技能。这样，就形成了三维空间结构

三维结构体系形象地描述了系统工程研究的框架。

时间维表示系统工程活动从开始到结束按时间顺序排列的全过程。分为七个时间阶段。

时间维几乎适用于任何工程，不论是产品生产过程还是软件生产过程，几乎都差不多。

逻辑维是指时间维的每一个阶段内所要进行的工作内容和应遵循的思维程序，包括七个逻辑步骤。逻辑维几乎覆盖了系统工程理论方法的各个方面。

最优化问题—从现实实例出发

- *与最大、最小、最长、最短等等有关的问题都是优化问题*
- 例1 作物种植安排
- 一个农场有50亩土地， 20个劳动力,计划种蔬菜,棉花和水稻. 种植这三种农作物每亩地分别需要劳动力 1/2 1/3 1/4, 预计每亩产值分别为 110元, 75元, 60元. 如何规划经营使经济效益最大.
- 分析：以取得最高的产值的方式达到收益最大的目标.

1. 求什么？分别安排多少亩地种蔬菜、棉花、水稻？ x_1 亩、x_2 亩、 x_3 亩
2. 优化什么？ 产值最大 $\max\ f=10x_1+75x_2+60x_3$
3. 限制条件？ 田地总量 $x_1+x_2+x_3 \le 50$ 劳力总数 $1/2x_1+1/3x_2+1/4x_3 \le 20$

最优化问题—从现实实例出发

- 例1 作物种植安排--对偶问题
- 一个农场有50亩土地，20个劳动力，计划种蔬菜,棉花和水稻. 种植这三种农作物每亩地分别需要劳动力 1/2 1/3 1/4, 预计每亩产值分别为 110元, 75元, 60元. 如何规划经营使经济效益最大.
- 分析：以最经济的投入达到收益最大的目标.（或者说以直接出售土地和劳动力的方式达到收益最大的目标.）
- 1 求什么？土地成本价格 y_1 劳动力成本价格 y_2
- 2. 优化什么？ 成本价格最低 Min $g=50y_1+20y_2$
- 3. 限制条件？
- 蔬菜的市场价 $y_1+1/2y_2 \geq 110$
- 棉花的市场价 $y_1+1/3y_2 \geq 75$
- 水稻的市场价 $y_1+1/4y_2 \geq 60$

思维模式不同！！

最优化问题—从现实实例入手

- 例3 钢材截短（切割问题）
- 有一批钢材，每根长7.3米. 现需做100套短钢材. 每套包括长2.9米, 2.1米,1.5米的各一根. 至少用掉多少根钢材才能满足需要, 并使得用料最省.

分析： 可能的截法和余料

第1种 7.3-(2.9× 2+1.5)=0
第2种 7.3-(2.9+2.1 × 2)=0.2
第3种 7.3-(2.9+1.5 × 2)=1.4
第4种 7.3-(2.9+2.1+1.5)=0.8
第5种 7.3-(2.1 × 2+1.5 × 2)=0.1
第6种 7.3-(2.1 × 3)=1
第7种 7.3-(2.1+1.5 × 3)=0.7
第8种 7.3-(1.5 × 4)=1.3

模型：设决策变量：按第i种方法截 x_i 根钢材。
求目标函数 $f=0.2x_2+1.4x_3+0.8x_4+0.1x_5+x_6+0.7x_7+1.3x_8$
在约束条件 $2x_1+x_2+x_3+x_4=100$ $2x_2+x_4+2x_5+3x_6+x_7=100$
$x_1+2x_3+x_4+2x_5+3x_7+4x_8=100$ $x_i \geq 0$, $i=1,\ldots,8$ 下的最小值

最优化的重要性

1. 人类的一切活动都是认识世界和改造世界的过程

即： 认识世界 → 改造世界

↓ ↓

(建模) (优化)

2. 一切学科都是建模与优化在某个特定领域中的应用

概念模型(定性) → 结构模型(图)→

→ 数学模型 → 智能模型

1.最优化的重要性

3. 最优化理论的发展

① 极值理论；

② 运筹学的兴起(Operation Research)；

③ 数学规划：线性规划(LP)；非线性规划(NLP)；动态规划(DP)；马尔可夫决策规划(MDP)；排队论；决策论；存储论。

4. 最优化理论在国民经济中的广泛应用

最优化问题---一般性描述

- 问题→（选择不同的）方法→（得到不同的）结果。
- 采取什么样的选择或行动最好，就是一个最优化问题。

$$\min \quad f(x)$$
$$s.t. \quad x \in D$$

- 优化问题的三要素：
  - 决策变量
  - 约束条件
  - 目标函数

- 所谓的优化问题，就是在某限制条件下（领域 **D**），从多种可行解中，确定使目标函数 $f(x)$ 达到最小（或最大）时，决策变量 x 的取值（最优解）问题。

优化问题--举例

- 背包问题(knapasck)
- 装箱问题(bin packing)
- 二次指派问题(QuadraticAssignment)
- 最小生成树问题(MinimumSpanningTree)
- 旅行商问题(TSP)
- 图着色问题(graph coloring)
- 聚类问题(clustering)
- 最优切割问题（cutting stock）
- 设备布局问题(layout)
- 作业、流水车间调度问题(flow shop, job-shop)
- 机器调度问题(scheduling)
- 物流、交通、运输问题(transportation)
- 多目标优化问题（multiobject）
- 模糊优化问题（fuzzy）
- ……

优化问题在日常生活中无处不在！

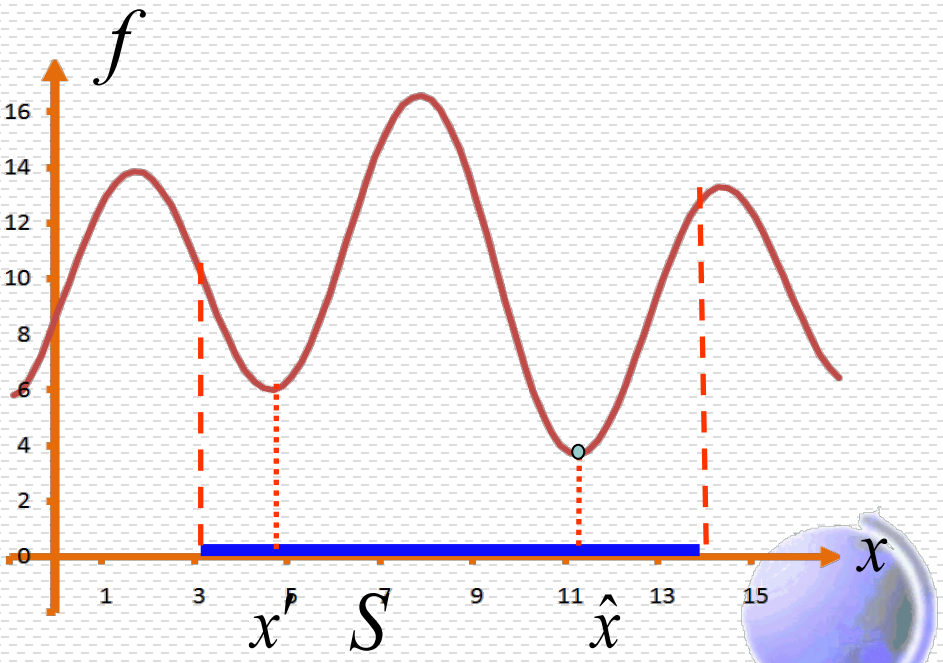
要买身西服，要考虑预算多少、今后的购入计划等，尽可能选择中意的东西。
要乘车出游，尽可能选择时间短、花钱少的路线，如果可能得话，还要选择沿途风景好的线路。
结婚办婚礼…
买房子装修…

最优化问题的简单实例（函数最优化）

- 函数优化问题：

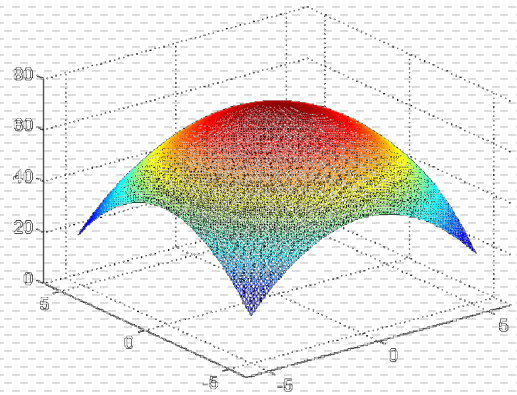
$$\min f(x) = 10 + 5\sin(x) + \log(10x + 11)\cos(3 + x/2)$$
$$s.t. \quad x \geq 3, \quad x \leq 14$$

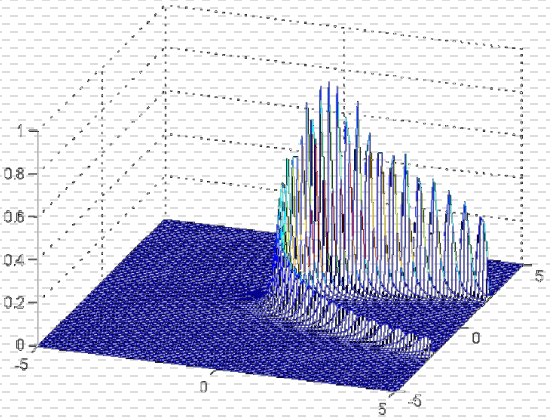
  - 目标函数：$f(x)$—〉最小化
  - 约束条件：$x \in S$
  - 最优值：$f(\hat{x})$
  - 最优解：$\hat{x}$
  - 局部最优解：x'
  - 多极值点、多峰问题

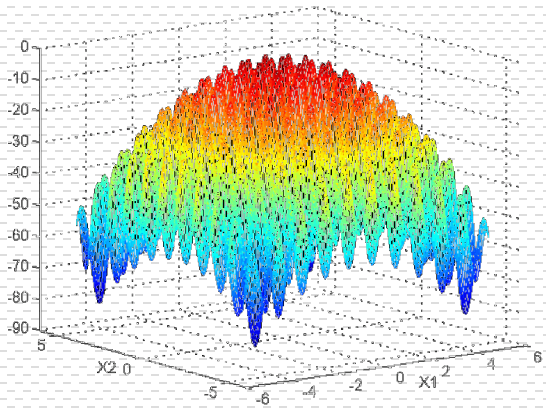


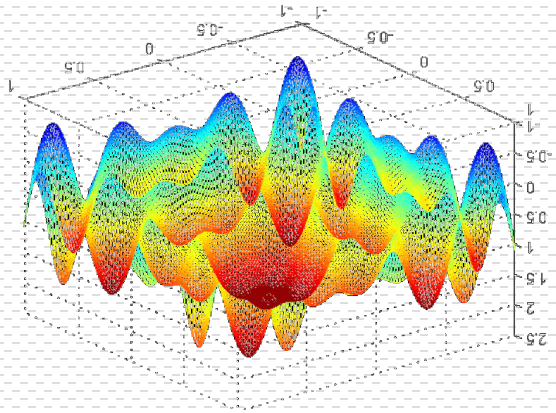
最优化问题的简单实例

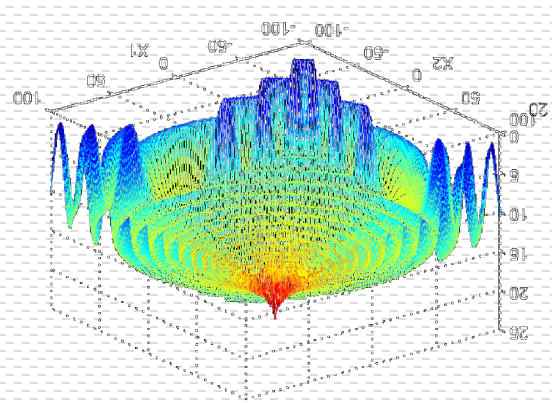
- 函数优化问题：

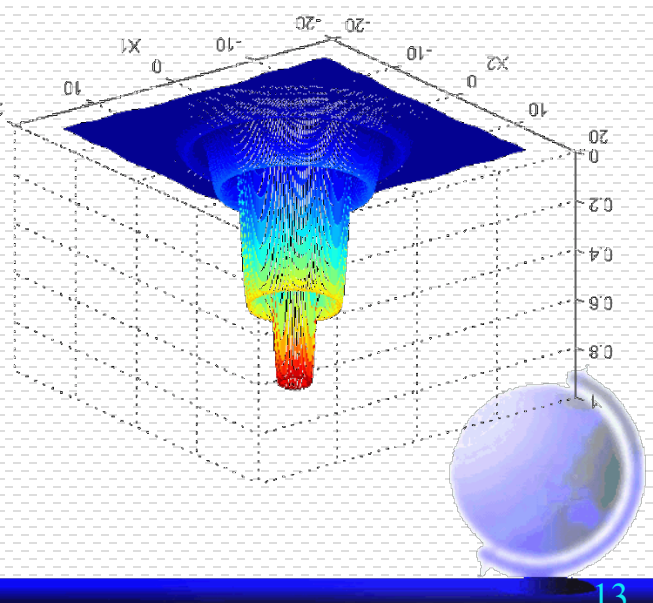






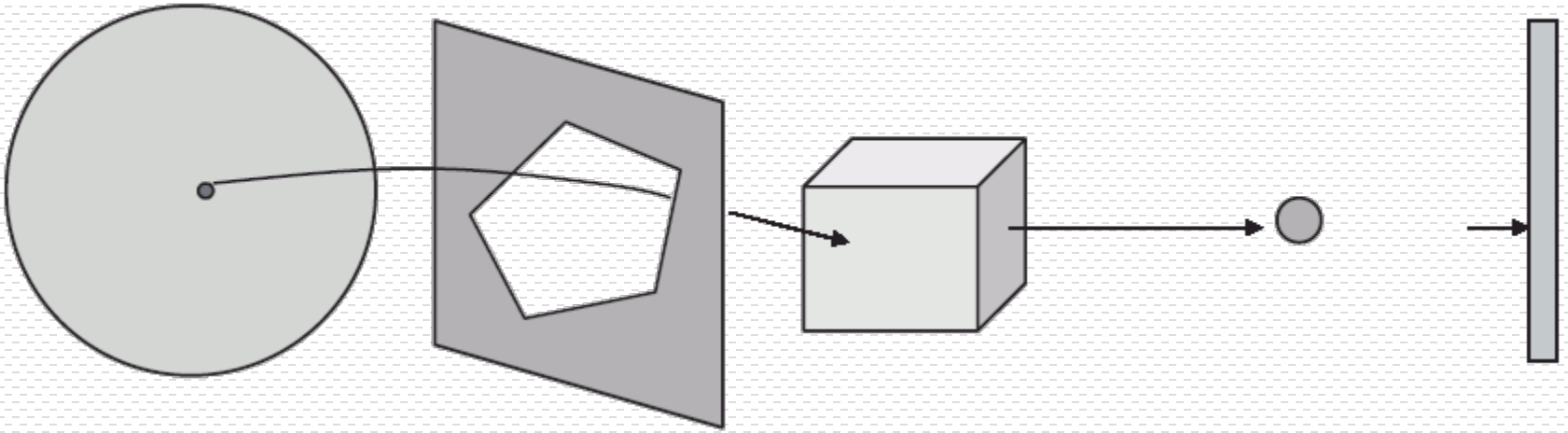






优化问题的基本结构

- 由5个部分构成



定义域：D
Domain

约束条件：C
Constraints

评价机制：M
Mechanism

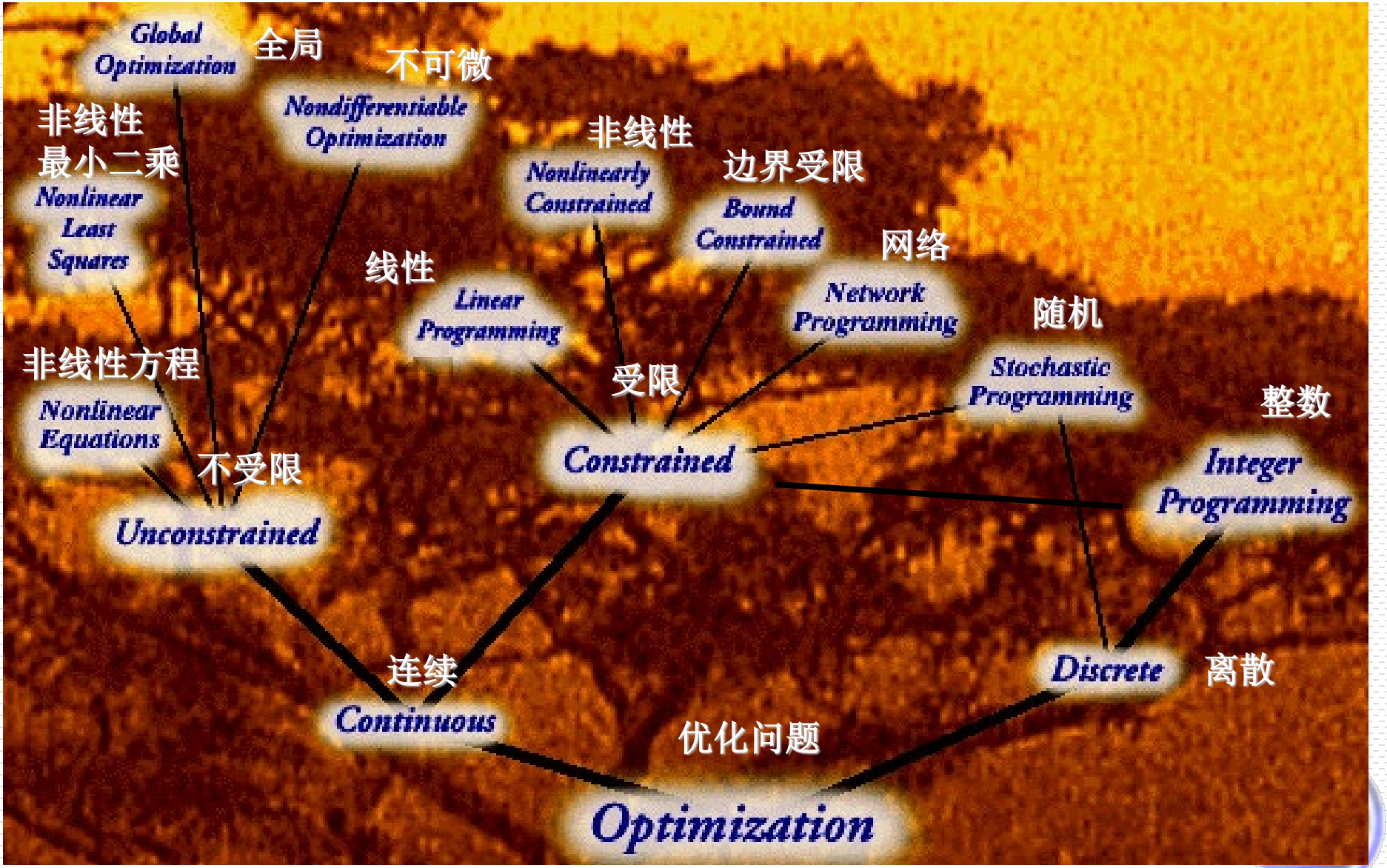
评价值：E
Value

最优性准则：G
Criterion

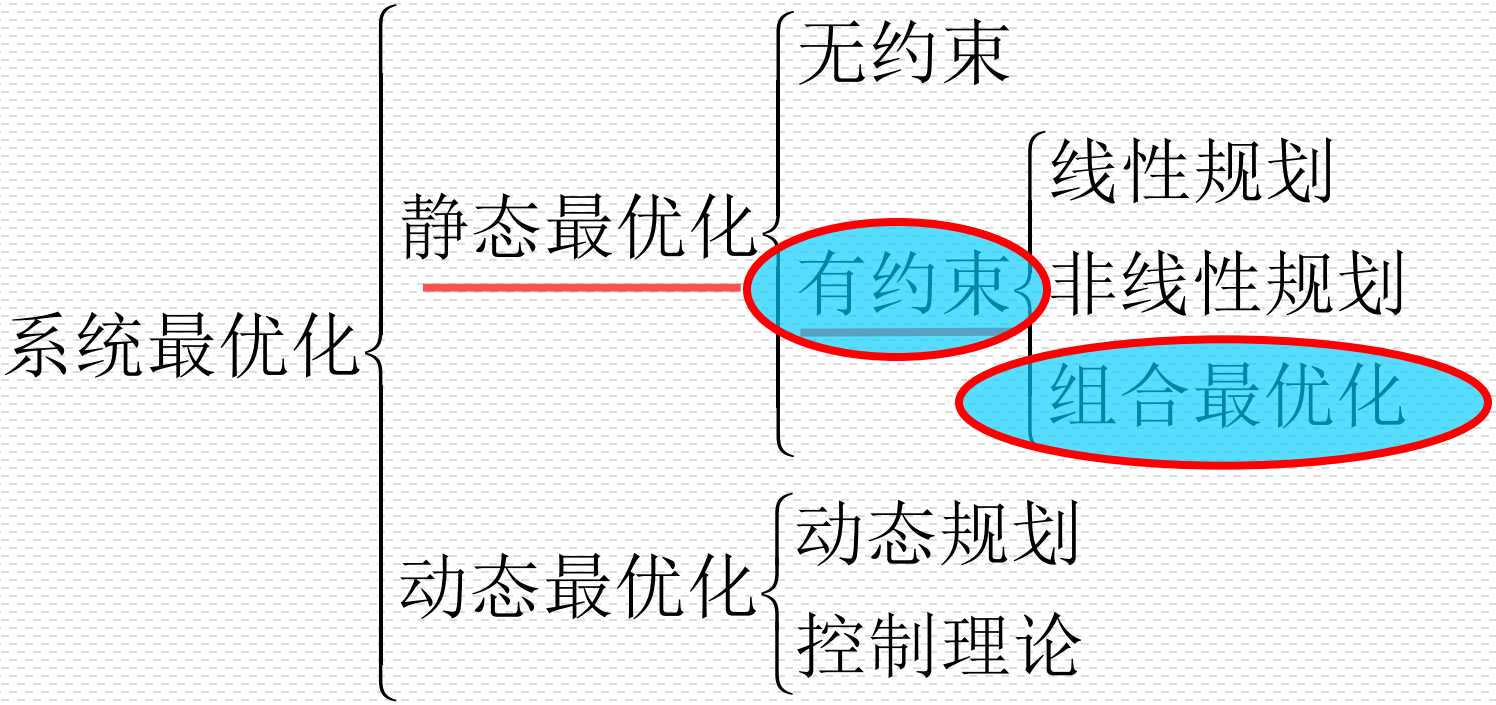
不同的组合，不同的优化问题

- D:实数，C:线性不等式，M:表达式，E:线性（标量），G:最大（小）
  - 线性规划 (linear programming problem)
- D:整数，C:线性不等式，M:表达式，E:线性（标量），G:最大（小）
  - 整数规划 (integer programming proglem)
  - 离散最优化 (discrete optimization problem)
  - 混合整数规划 (mixed integer programming problem)
  - 0-1整数规划、混合0-1整数规划
- D:实数，C:线性不等式，M:表达式，E:非线性（标量），G:最大（小）
  - 二次规划（quadratic programming problem)
  - 非线性规划（non-linear programming problem)
- D:函数，C:边界条件、连续性等，M:表达式，E:泛函，G:最大（小）
  - 变分问题(variational problem)
- D:对象的组合，C:各种条件，M:一般为表达式，E:一般为标量，G:最大（小）
  - 组合最优化问题(combinational optimization problem)
    - 例如:旅行商问题(traveling salesman problem)

优化问题的分类


Global Optimization
全局
不可微
Nondifferentiable Optimization
非线性
最小二乘
Nonlinear Least Squares
非线性
Nonlinearly Constrained
边界受限
Bound Constrained
网络
Network Programming
线性
Linear Programming
随机
Stochastic Programming
非线性方程
Nonlinear Equations
受限
Constrained
整数
Integer Programming
不受限
Unconstrained
连续
Continuous
Discrete
离散
优化问题
Optimization

不同的分类方法



无约束优化问题

(Unconstraint Optimization Problems)

多峰函数测试函数--Ackley函数

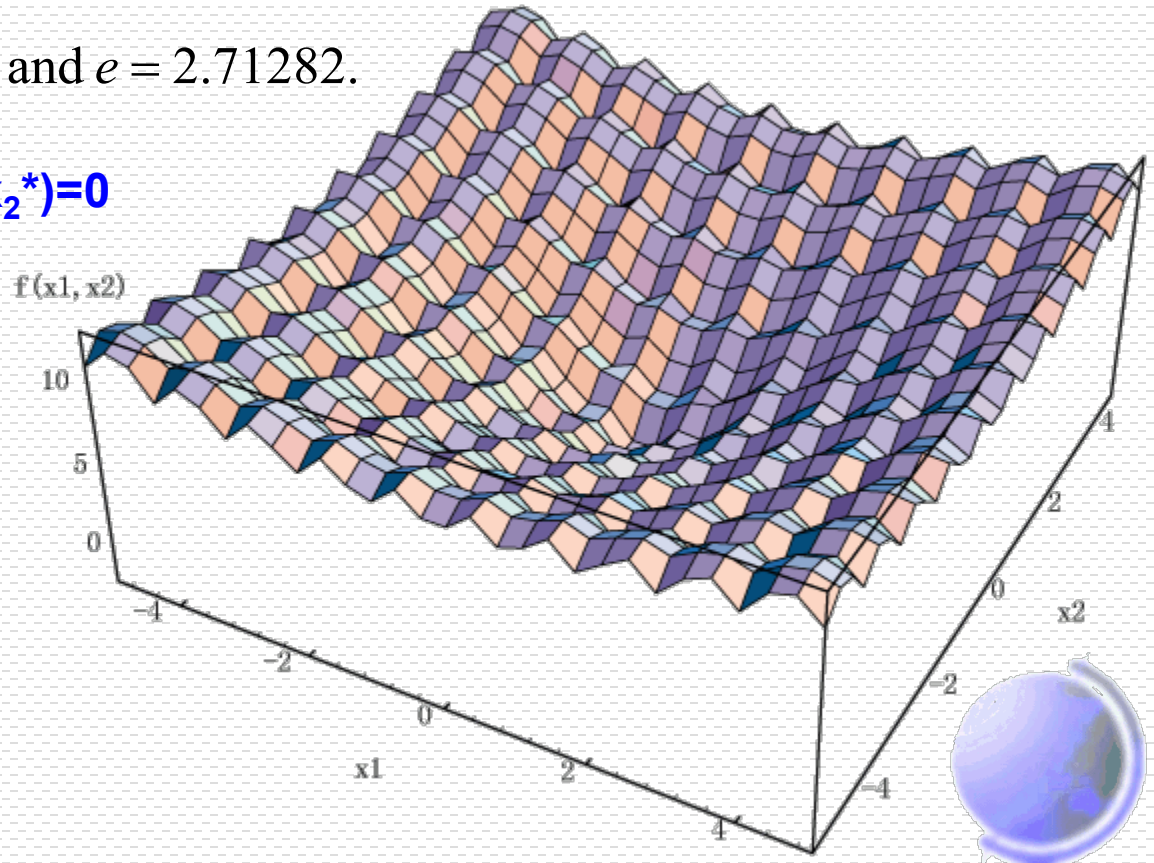
- 多峰的一个测试函数

$$\min f(x_1,x_2) = -c_1 \cdot \exp\left(-c_2\sqrt{\frac{1}{2}\sum_{j=1}^{2}x_j^2}\right) - \exp\left(\frac{1}{2}\sum_{j=1}^{2}\cos(c_3 \cdot x_j)\right) + c_1 + e$$

$$-5 \le x_j \le 5, \quad j = 1,2$$

where $c_1 = 20, c_2 = 0.2, c_3 = 2\pi$, and $e = 2.71282$.

最优解: $(x_1^*, x_2^*)=(0, 0)$, $f(x_1^*, x_2^*)=0$



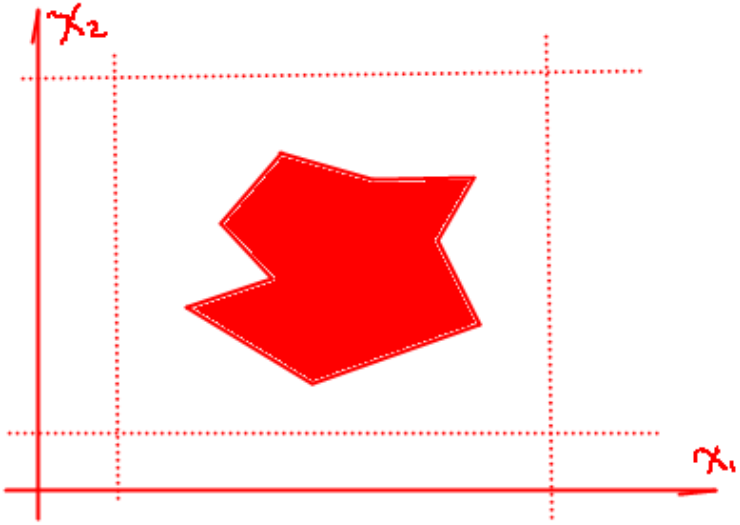
约束优化问题-非线性规划问题

(Nonlinear Programming Problems)

非线性规划问题

- 在等式和/或不等式约束的前提下，优化某个目标函数的问题，称为非线性规划。
- 由于许多实际问题不能成功地表述为线性规划模型，因此，非线性规划对于工程、数学和运筹学的各个领域都是极其重要的工具。
- 一般非线性规划可描述如下：

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s. t.} \quad & g_i(\boldsymbol{x}) \le 0, \quad i = 1,2,\ldots,m_1 \\ & h_i(\boldsymbol{x}) = 0, \quad i = m_1+1,\ldots,m_1+m_2 \\ & \boldsymbol{x} \in \boldsymbol{X} \end{aligned}$$



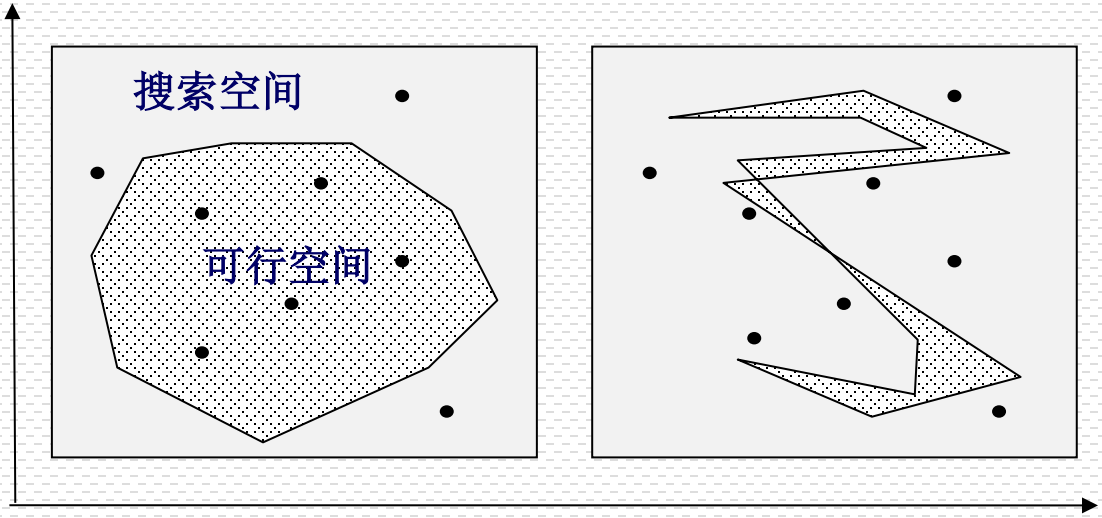
- 非线性规划问题就是寻找一点 y 使得对任意的可行解x 都存在 $f(y) \le f(x)$
- 和线性规划不同,传统的非线性规划方法十分复杂且效率不高。

约束条件的处理技术

- 几种处理约束的技术：
  - 拒绝策略 (Rejecting Strategy);
  - 修复策略 (Repairing Strategy);
  - 改进遗传算子策略 (Modifying Genetic Operator Strategy);
  - 惩罚策略 (Penalty Strategy)。
- 各种策略都有不同的优点和缺点。

约束条件的处理技术--拒绝策略

- 拒绝策略抛弃所有遗传过程中产生的不可行的染色体。
- 这是遗传算法中普遍的作法。
- 当可行的搜索空间是凸的，且为整个搜索空间的适当的一部分时，这种方法应该是有效的。
- 然而，这是很严格的限制。
  - 例如，对许多约束优化问题初始种群可能全由非可行染色体构成，全部拒绝会导致算法无法运行。

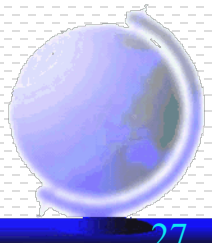


约束条件的处理技术--修复策略

- 修复策略对不可行染色体采用“修复程序”使之变为可行。
  - 对于某些系统，特别是可行搜索空间非凸时，通过对不可行染色体的修复，往往更容易达到最优解。
  - 对于许多组合优化问题，构造修复程序相对比较容易。
  - 能否采取修复策略，取决于是否存在一个可将不可行染色体转化为可行染色体的修复程序。
- 该方法的缺点是：
  - 它对*问题本身的依赖性*，*对于每个具体问题必须设计专门的修复程序*。
  - 对于某些问题，*修复过程甚至比原问题的求解更复杂。*
- 最近，Orvosh和Davis提出了所谓的5%规则：
  - 该规则对于多数组合优化问题，若令5%的修复过的染色体替代原染色体，则带有修复程序的遗传算法可取得最好的效果。
- Michalewicx 等则认为对有非线性约束的优化问题，15%的替代律为最好。

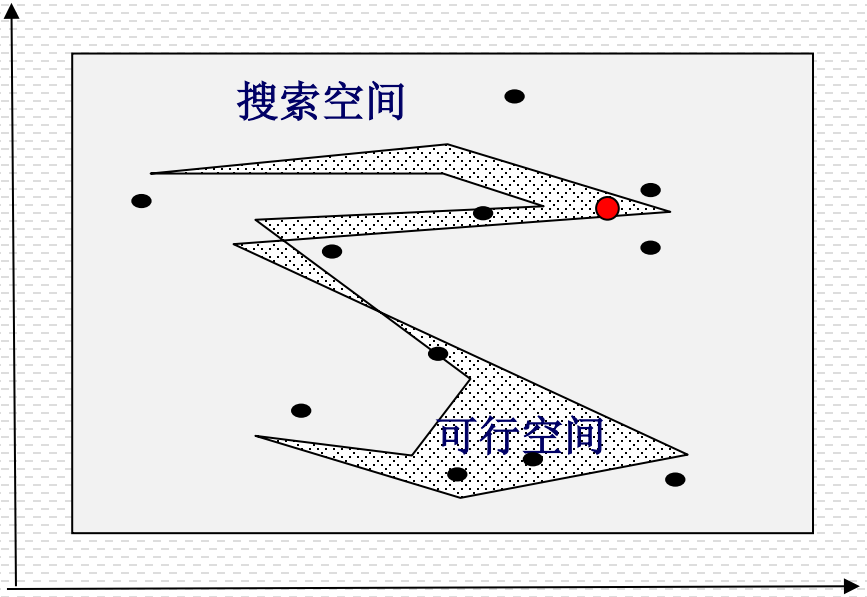
约束条件的处理技术--改进遗传算子策略

- 解决可行性问题的一个合理办法是设计针对问题的表达方式以及专门的遗传算子来维持染色体的可行性。
- 许多领域中的实际工作者采用“专门的问题表达方式和遗传算子”构造了非常成功的遗传算法，这已是一个十分普遍的趋势。
- 这种方法通常比基于惩罚的遗传算法更可靠。
- 但是，该方法的遗传搜索被限制在可行域内。无法搜索可行域外的点。



约束条件的处理技术--惩罚策略

- 对于约束严的问题，不可行解在种群中的比例很大。如果将搜索限制在可行域内，就很难找到可行解。
- **Glover 和 Greenberg 建议的约束处理技术：**
  - 允许在搜索空间的不可行域中进行搜索，要比将搜索限制在可行域内的方法能更快地获得最优解或较好解。
- 惩罚策略就是这类*在遗传搜索中考虑不可行解的技术*。



约束条件的处理技术--惩罚策略

- 惩罚策略本质上是通过惩罚不可行解，将约束问题转化为无约束问题。
  - 例如，对如下约束优化问题：

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s. t.} \quad & h(\boldsymbol{x}) = 0 \end{aligned}$$

通过乘子转化为无约束优化问题：

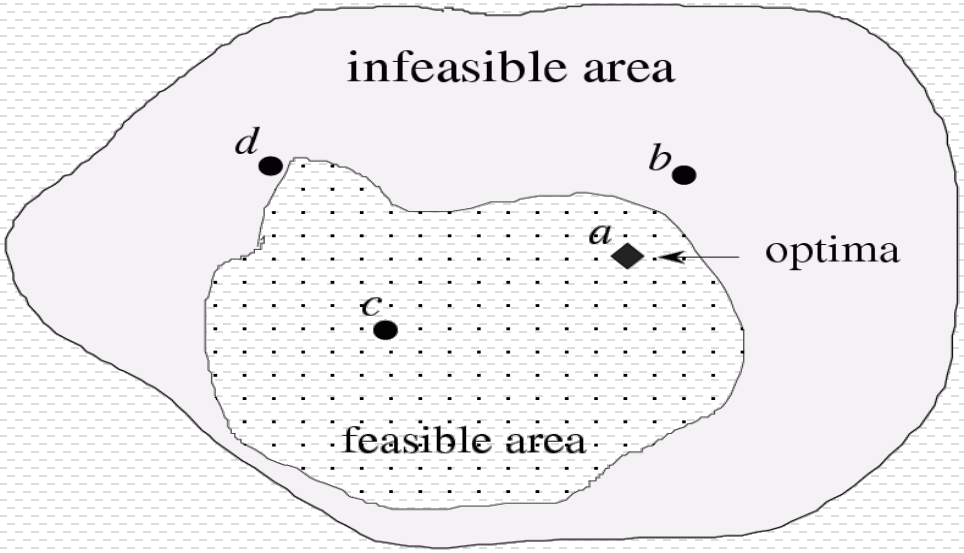
$$\min \quad f(\boldsymbol{x}) + \lambda \left| h(\boldsymbol{x}) \right|$$

将不可行解对约束的违反程度作为惩罚项加入目标函数

  - 对 GA 而言，这种惩罚反应在适值函数上，通过对违反约束的染色体进行惩罚，使之具有较低的适值。

约束条件的处理技术--惩罚策略

- 惩罚技术用来在每代的种群中保持部分不可行解，使遗传搜索可以从可行域和不可行域两边来达到最优解
- 要想保留不可行解，惩罚必须适当，否则要么被淘汰、要么最后找不到可行解。



- 惩罚策略的主要问题是：
  - 如何设计一个惩罚函数 $P(X)$，从而能有效地引导遗传搜索达到解空间的最好区域。

约束条件的处理技术--惩罚函数

- 设计惩罚函数没有一般规则，仍要依赖于待解的问题。
- 带惩罚项的适值函数
  - 加法形式：

$$eval(\boldsymbol{x}) = f(\boldsymbol{x}) + p(\boldsymbol{x})$$

$$p(\boldsymbol{x}) = 0 \quad \boldsymbol{x} \text{ 可行解}$$

$$p(\boldsymbol{x}) < 0 \qquad \text{其它}$$

  - 乘法形式：

$$eval(\boldsymbol{x}) = f(\boldsymbol{x}) \cdot p(\boldsymbol{x})$$

$$p(\boldsymbol{x}) = 1 \quad \boldsymbol{x} \text{ 可行解}$$

$$0 \le p(\boldsymbol{x}) < 1 \quad \text{其它}$$

假设问题是求极大。这里 $\boldsymbol{x}$ 表示一个染色体，$f(\boldsymbol{x})$ 表示目标函数，$p(\boldsymbol{x})$ 表示惩罚项。

约束条件的处理技术--惩罚函数

- 惩罚函数的分类
  - 基本上，惩罚是距可行域的距离的函数。这个距离可按以下三种方式测量:
    - 单一不可行解的绝对距离的函数;
    - 现行种群中所有不可行解的相对距离的函数;
    - 自适应惩罚项的函数。
  - 一般可分为两类
    - 定量惩罚
      - 不考虑约束违反程度，惩罚量是一定的。
      - 定量惩罚法，对于复杂问题不太有效
    - 变量惩罚。一般包含两部分:
      - 可变惩罚率：可按约束违反程度和 GA 的迭代次数进行调节。
      - 违反约束的惩罚量.

约束条件的处理技术--惩罚函数

- 惩罚函数的分类
  - 惩罚法可进一步分为
    - 问题依赖的;
      - 多数惩罚技术属于问题依赖的
    - 问题独立的。
  - 惩罚法还可分为
    - 带参数的;
      - 多数惩罚技术属于带参数的，
      - 参数惩罚函数基本都是问题依赖的
    - 不带参数的。

典型惩罚函数（1）

- **Homaifar，Qi 和 Lai 方法**

考虑如下非线性规划问题

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geqslant 0;\ i = 1,2,\cdots,m \end{aligned}$$

取加法形式的惩罚函数

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x})$$

$$p(\mathbf{x}) = \begin{cases} 0, & \text{若 } \mathbf{x} \text{ 可行} \\ \sum_{i=1}^{m} r_i g_i(\mathbf{x}), & \text{其他} \end{cases}$$

其中 r_i 是约束 i 的可变惩罚系数。

仅包含违反约束的项（违反约束为负数），而不是全部约束

- 惩罚函数由两部分构成:
  - 可变惩罚因子
  - 违反约束惩罚。
- MIichalewicz最近指出，解的质量严重地依赖于这些惩罚系数的值。当惩罚系数不适当时，算法可能收敛于不可行解;另一方面，惩罚系数过大时该方法等价于拒绝策略。

典型惩罚函数（2）

- **Joines 和 Houck 方法**

考虑如下非线性规划问题

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geqslant 0;\ i = 1,2,\cdots,m_1 \\ & h_i(\mathbf{x}) = 0;\ i = m_1+1,2,\cdots,m\ (=m_1+m_2) \end{aligned}$$

取加法形式的评估函数：

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(t,\mathbf{x})$$

$$p(t,\mathbf{x}) = -\rho_t^{\alpha}\sum_{i=1}^{m} d_i^{\beta}(\mathbf{x})$$

$$d_i(\mathbf{x}) = \begin{cases} 0, & \text{若 } \mathbf{x} \text{ 可行} \\ |g_i(\mathbf{x})|, & \text{否则 } 1 \leqslant i \leqslant m_1 \\ |h_i(\mathbf{x})|, & \text{否则 } m_1+1 \leqslant i \leqslant m \end{cases}$$

$$\rho_t = C \times t$$

仅包含违反约束的项，而不是全部约束

t 是遗传算法的迭代次数，α 和 β 是调节惩罚值大小的参数。C 是常数。通过 ρ_t 对不可行染色体的惩罚随进化过程进展而增加。

- 惩罚函数由两部分构成:
  - 可变惩罚因子
  - 违反约束惩罚。
- 可变惩罚因子随迭代次数而变。解的质量对三个参数的值很灵敏。
- 在遗传算法迭代的后期，该方法将给不可行染色体以死亡惩罚。该方法会因惩罚过大而过早收敛。

典型惩罚函数（3）

- **Michalewicz 和 Attia 方法**

考虑如下非线性规划问题

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geqslant 0;\ i = 1,2,\cdots,m_1 \\ & h_i(\mathbf{x}) = 0;\ i = m_1+1,\cdots,m(=m_1+m_2) \end{aligned}$$

取加法形式的评估函数：

$$eval(\mathbf{x}) = f(\mathbf{x}) + p(\tau,\mathbf{x})$$

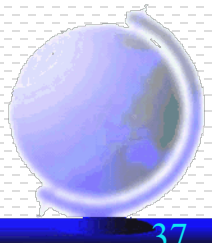
$$p(\tau,\mathbf{x}) = -\frac{1}{2\tau}\sum_{i\in A} d_i^2(\mathbf{x})$$

$$d_i(\mathbf{x}) = \begin{cases} \min\{0, g_i(\mathbf{x})\}, & \text{若 } 1 \leqslant i \leqslant m_1 \\ |h_i(\mathbf{x})|, & \text{若 } m_1+1 \leqslant i \leqslant m \end{cases}$$

A 是起作用的约束集，它由所有等式约束和不能满足的不等式约束构成。τ 是可变惩罚因子，称为温度。从初始温度 τ_0 开始，终止在冻结温度 τ_f

- 惩罚函数由两部分构成：
  - 可变惩罚因子
  - 违反约束惩罚。
- 此方法对参数值十分灵敏。存在的问题是对具体问题应如何设定参数。

系统建模与数学模型

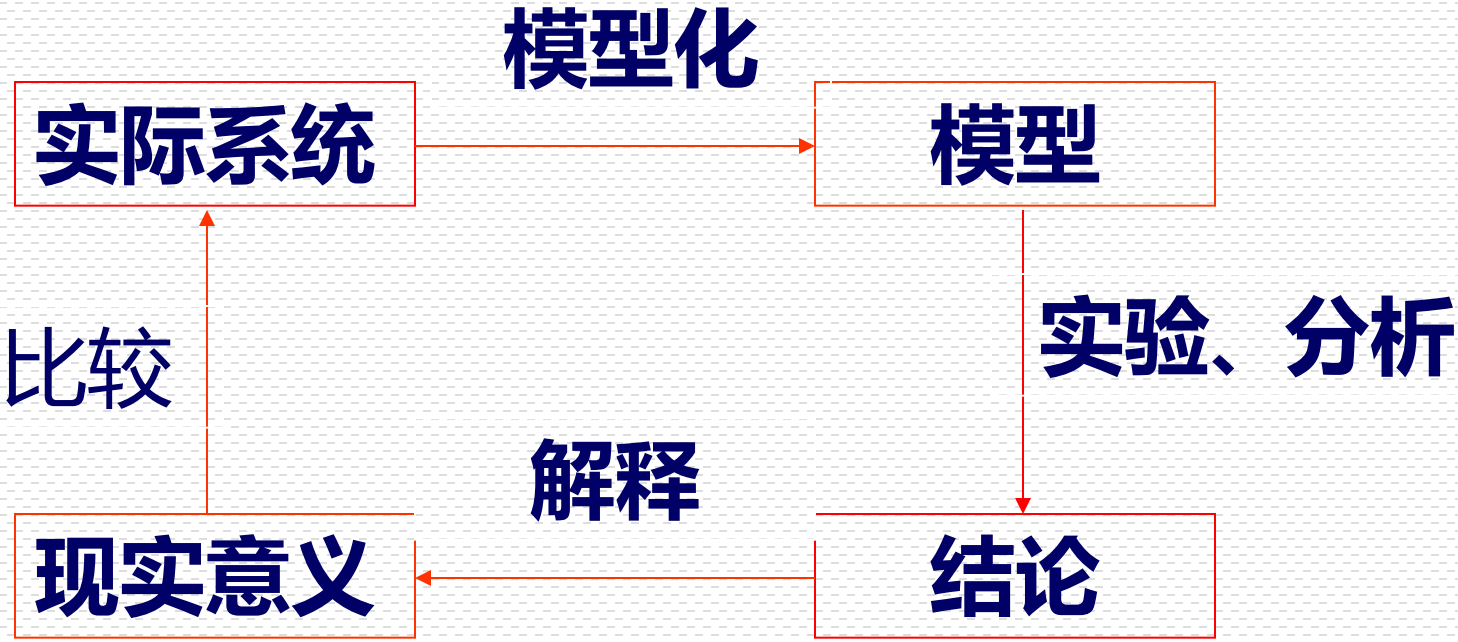


系统建模与数学模型

■ 有关概念

- **系统建模**
  - ✓ 对于所研究的系统通过一些工具、手段和方法（如类比、模拟或抽象）建立模型的过程；
  - ✓ 系统建模的目的是根据总体最优化的目标，对系统进行优化；
- **模型**
  - ✓ 现实世界或问题的一种表达形式，是客观世界经过抽象思维后某种符号、文字、图表、关系以及实体模样的一种表达

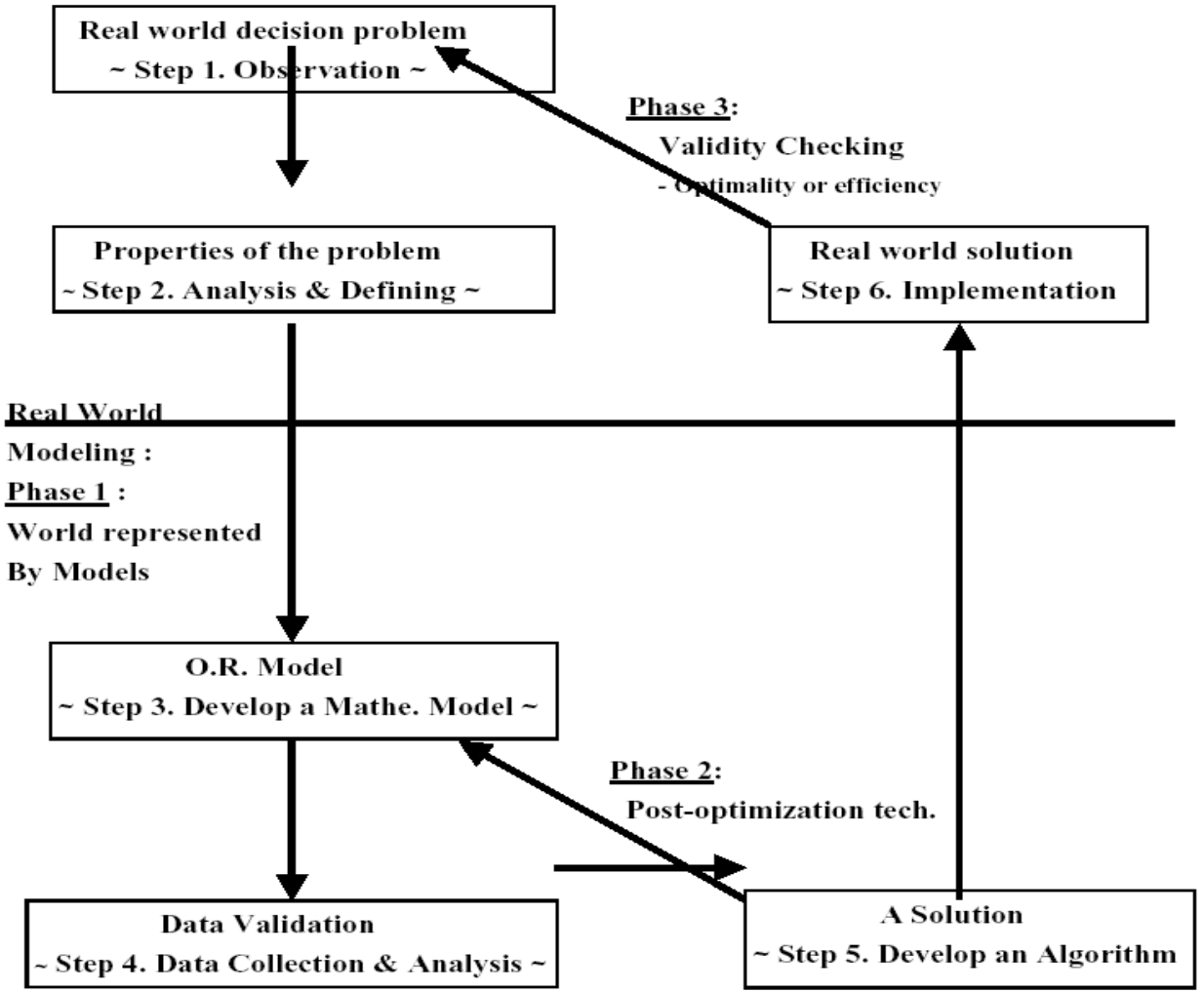
系统建模与数学模型



系统模型（化）的作用与地位

系统建模与数学模型

- The Scientific Methodology of Operations Research



- 模型的分类 （表征特征）

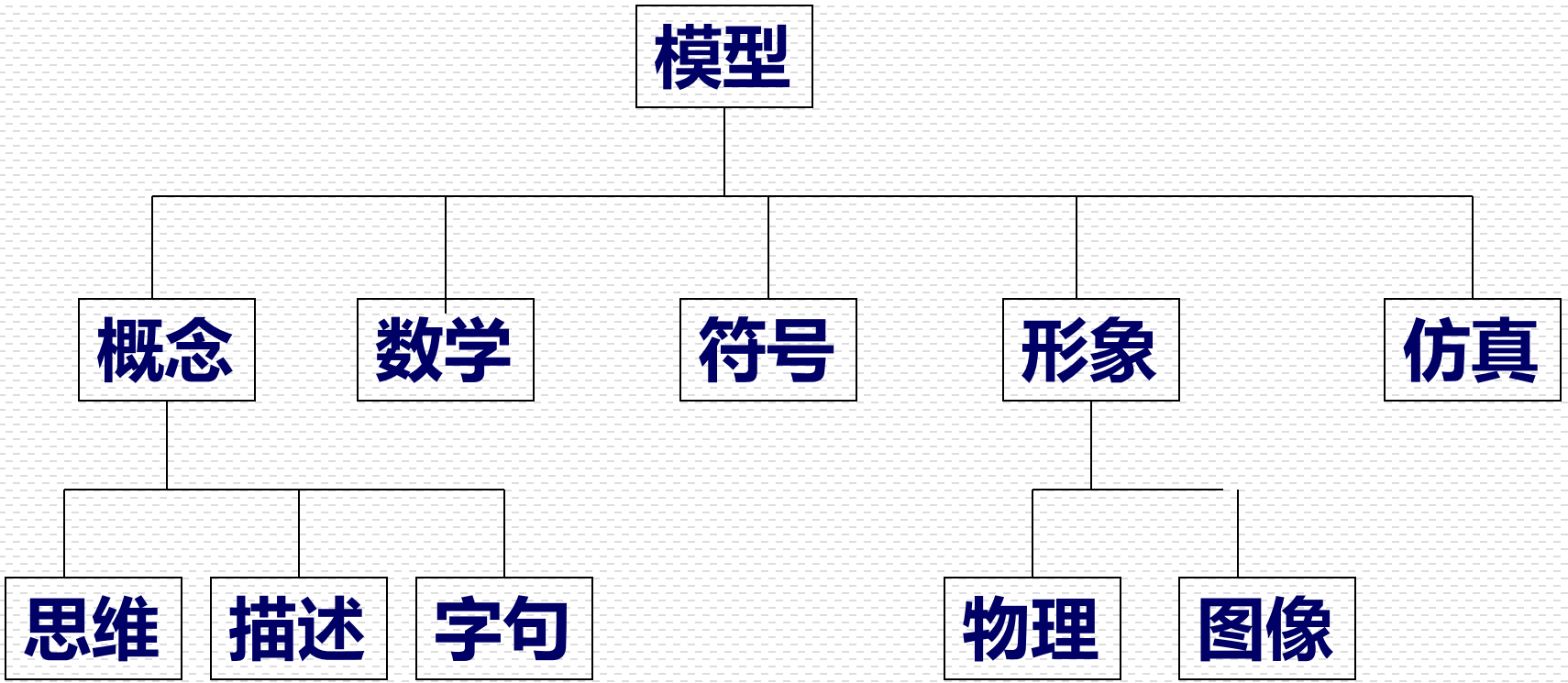


图2 模型分类

系统建模与数学模型

■ 模型的分类

- **按总体观点分类**

  物理模型与思考模型

- **按运动特性分类**

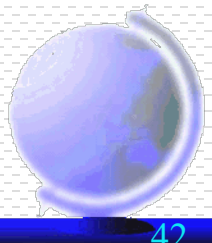
  静态模型与动态模型

- **按用途分类**

  预测模型、决策模型、对策模型、功能模型、规划模型、评估模型、投入产出模型

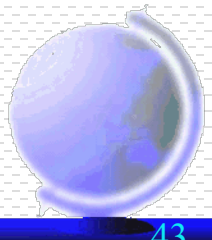
- **按变量特征分类**

  连续模型、离散模型、混合模型；确定型模型、非确定型模型



模型的分类

- 按建立模型的数学方法分类
  - 几何模型 、微分方程模型、图论模型 、规划模型...
- 按人们对事物发展过程的了解程度分类
  - 白箱模型：
    - 指那些内部规律比较清楚的模型。如力学、热学、电学以及相关的工程技术问题。
  - 灰箱模型：
    - 指那些内部规律尚不十分清楚，在建立和改善模型方面都还不同程度地有许多工作要做的问题。如气象学、生态学、经济学等领域的模型。
  - 黑箱模型：
    - 指一些其内部规律还很少为人们所知的现象。如生命科学、社会科学等方面的问题。但由于因素众多、关系复杂，也可简化为灰箱模型来研究。



系统建模与数学模型

■ 建模的基本步骤

➢**模型问题化。**明确建模的目的（说明解决问题、预测、决策和设计新的系统）、对象（自然科学、社会科学、工程技术领域）和要求（模拟、仿真；定性、定量结果表达）；包括解决哪些问题、如何解决这些问题；以便使模型满足实际要求，不致产生太大偏差；

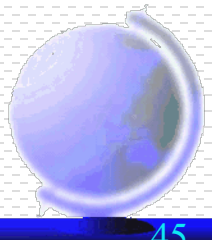
➢ **模型目标化。**建立模型的目标，如质量、效益、成本等；并表述为相应的形式，如单目标、多目标；最大化、最小化；

➢ **模型规范化。**如问题的界限、解决问题的方式要求、精度要求、结果的表达形式等

系统建模与数学模型

■ 建模的基本步骤 (续)

- **模型要素化。**弄清系统中的主要因素（变量）及其相互关系（结构关系和函数关系），以便使模型准确表示现实系统；
- **模型关系化。**确定模型的结构，分析模型各要素之间的影响和因果联系，包括约束条件等；奠定模型量化的基础；
- **模型参数化。**估计模型的参数，用数量来表示系统中的因果关系；
- **模型形式化。**选择模型的某种表达形式，如定量化表示；
- **模型简洁化。**在反映模型问题、模型目标和模型规范为前提下，对模型进行进一步的简化；
- **模型求解**
- **模型检验与修订**



系统建模与数学模型

■ 概述

➢ 概念

利用数学语言来描述系统的行为的一类模型

➢ 变量性质分类：

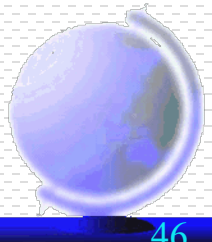
确定性模型、随机模型、模糊模型、灰色模型；连续模型、离散模型、混合模型；

➢ 线性模型与非线性模型；

➢ 动态模型与静态模型；

➢ 系统工程与管理科学领域中的常用模型

经济模型、人口模型、交通模型、生态模型、预测模型、决策模型、排队模型、库存模型、规划模型、评估模型等



■ 数学建模的一般考虑

- 了解系统属于哪个领域、与哪些理论或有关学科
- 要建立哪种类型的数学模型
- 要建立哪种关系的数学模型：逻辑关系、输入输出关系、集合关系、混合关系型
- 要考虑系统中的哪些因素及因素间的关系：成分、变量、参数；
- 要考虑建模系统与环境之间的关系；
- 考虑建模方法的选取，如演绎法、归纳法，综合统计数据的处理、测验与仿真等手段；

系统建模与数学模型

- 数学建模的主要步骤
  - 准备阶段：弄清问题、目标与目的；
  - 系统认识阶段：构成、环境、界限、约束等
  - 系统建模阶段：模型假设、建立关系式
  - 模型求解阶段：
  - 模型检验；

算法与优化算法概述

算法与优化算法——算法的概念

- 在现代数学中，任何一个一般性的*计算方法和计算程序*均可称之为算法。
- 一切可设想的数学问题其算法均可分两大类：
  - 一是无算法；
  - 一是有算法。
- 在有算法的问题中又分为：
  - 有效算法；
    - 是指计算时间只随问题规模的增大呈多项式曲线增长
  - 无效算法。
    - 是指计算时间随问题规模的增大呈指数曲线增长

这里的计算时间通常是指计算步骤。

算法与优化算法——算法举例

- **有效算法；**
  - 例如：对 n 个数字进行排列的选择排序算法，
  - 比较次数（n-1)+(n-2)+...+1 = n(n-1)/2,
  - 算法复杂度：$O(n^2)$，n 是问题的规模
  - n 由100增加到103，计算比较次数增加：
    (103*(103-1)/2) / (100*(100-1)/2)= 1.06倍
- **无效算法。**
  - 例如：工厂的下料问题，要在一块很大的钢板上切割出大小形状不同的钢片，怎么排列钢片才能用料最省。
  - 假设有 n 个钢片，采用枚举的方法有n!种可能，要进行n!次比较才能找到最佳下料组合， n是问题的规模
  - n 由100增加到103，计算比较次数增加：
    103！/ 100！ = 1百多万倍

这里的计算时间通常是指计算步骤。

算法与优化算法

- **所谓优化算法，实质就是一种*搜索过程*或*规则*。**
  - **基于某种思想或机制，通过一定的途径或规则得到满足用户要求的问题的解的过程。**

优化算法的分类—按优化机制与行为划分

- 经典算法：运筹学中的传统算法，计算复杂性一般很大，多数算法只适于求解小规模问题。
  - 线性规划
  - 动态规划
  - 整数规划
  - 分支定界法
- 构造算法：用构造的方法快速建立问题的解，通常算法的质量差，难以满足工程的要求，如调度中的
  - Johnson法(约翰逊)；SPT, LPT
- 迭代算法：从问题的任一解出发，对其邻域不断搜索并替换当前解，通过迭代来实现优化。
  - 邻域搜索法
  - 贪婪搜索（只接受优于当前解的邻域搜索方法）
  - 全局搜索(SA, GA, TS, EP)（利用一些“指导规则”在整个解空间中寻找优良解，可以接受坏的解）
- 演化算法：将优化过程转化为系统动态的演化过程，基于系统的动态演化来实现优化
  - 神经网络
  - 混沌搜索
- 混合算法：上述算法在结构或操作上相混合而产生的各种算法。

优化算法的分类-II

- 解决一个实际问题时，通常要问的两个问题：
  - 是必须求取全局最优解、还是局部最优解就可以满足要求？
  - 是必须获取精确解、还是近似解(准最优解)就可以满足要求？
- 精确解的求解算法(*经典算法*、确定算法)
  - 面向连续函数的方法
  - 面向离散函数的方法
- 近似解的求解算法
  - 随机法(random method)、 (Monte Carlo Method)
  - 邻域搜索法(neighborhood search)
  - 松弛法(relaxation method)

- 理想的情况是求取全局最优的精确解，然而对于大规模问题，能获取全局精确解的实际上不是很多，对于大多数优化问题，我们不得不妥协于求解局部最优的近似解。

优化问题的复杂性

- 有些问题往往是：
  - 不连续、不可微、多峰、随机、模糊、不确定、甚至无法用数学公式来进行描述的。
- 很多问题找不到精确的求解方法
  - 即使可以找到精确的求解方法，但随着问题规模的扩大，也无法在有限的时间（步骤）内得到问题的最优解。也就是找不到求解问题的多项式算法，
  - 属于一类NP难（或NP完全）的问题。
    - P(Polynomial)问题：可以找到多项式时间算法的问题。如果一个算法，它能在以输入规模为参变量的某个多项式的时间内给出答案，则称它为多项式时间算法。注意：这里的多项式时间是指算法运行的步数。
    - NP(Non-Deterministic Polynomial)问题：多项式复杂程度的非确定性问题。是指可以在多项式时间里验证一个解的问题。
    - NP完全(NP-complete)：可以规约为一类特殊的NP问题，这类问题目前还没找到多项式算法，但也没法证明找不到。
    - NP难(NP-hard)：比NP完全问题的范围更广，目前没有多项式算法（不一定是NP问题）。

传统优化方法的基本步骤与局限性

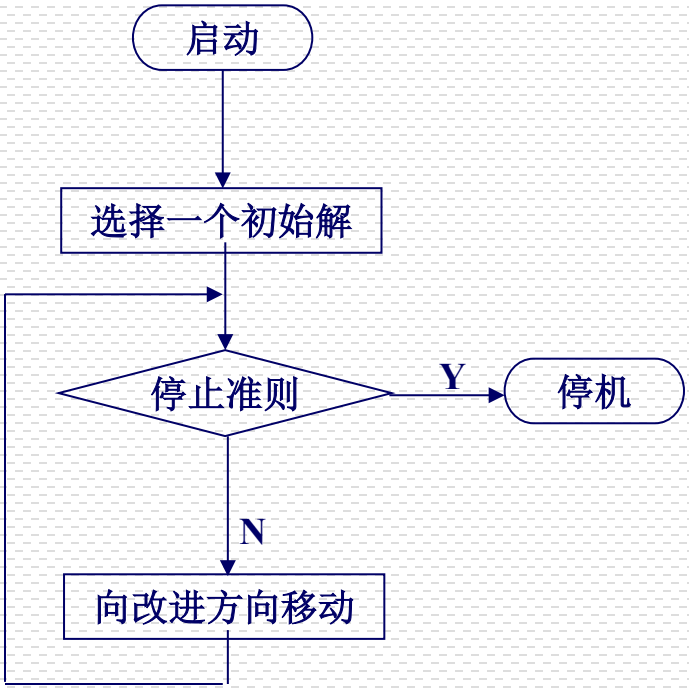
传统优化方法的基本步骤—三步曲

如下面框图所示

1. 选一个初始解

① LP:大M，二阶段法

② NLP:任意点或一个内点



传统优化方法的基本步骤与局限性

■ 传统优化方法的基本步骤—三步曲

- 2 停止判据——停止规则最优性检验
- LP:检验数 $\Pi = C_B^T B^{-1} N - C_N^T$

$A = [B \mid N]$ $C = [C_B \mid C_N]^T$

当∏≥0时有可能减小 Max Min?

- NLP: $\nabla f(x) = 0$

3. 向改进方向移动——改进解

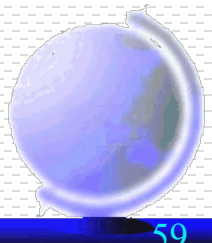
① LP:转轴变换(进基、退基)

② NLP:向负梯度方向移动(共轭梯度方向、牛顿方向)

传统优化方法的基本步骤与局限性

■ 传统优化方法的局限性

1. 对问题中目标函数、约束函数有很高的要求——有显式表达，线性、连续、可微，且高阶可微;
2. 只从一个初始点出发，难以进行并行、网络计算，难以提高计算效率;
3. 最优性达到的条件太苛刻——问题的函数为凸，可行域为凸;
4. 在非双凸条件下，没有跳出局部最优解的能力。

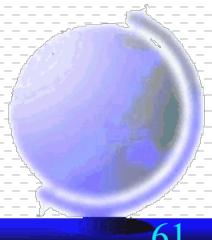


■ 实际问题中对最优化方法的要求

1. 对问题的描述要宽松(目标和约束函数)——
   可以用一段程序来描述(程序中带判断、循环)，函数可以非连续、非凸、非可微、非显式;
2. 并不苛求最优解——通常满意解、理想解就可以了;
3. 计算快速、高效,可随时终止(根据时间定解的质量);
4. 能够处理数据、信息的不确定性(如数据的模糊性，事件的随机性)。

智能优化方法概述

- 遗传算法
- 模拟退火算法
- 禁忌搜索算法
- 蚁群算法
- 粒子群算法
- 人工神经网络
- 混合智能优化算法
- 人工智能
- 机器学习、深度学习、随机森林……



智能优化方法概述-遗传算法

- 遗传算法（Genetic Algorithm）是一类*借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）*演化而来的随机化搜索方法。
- 由美国的J.Holland教授1975年首先提出
- 其主要特点：
  - 遗传算法运算的是解集的编码，而不是解集本身；
  - 搜索始于解的一个种群（多点），而不是单个解，具有内在的隐并行性和更好的全局寻优能力；
  - 只使用报酬信息（适值函数），而不使用导数或其它辅助知识；
  - 采用概率的，而不是确定的状态转移规则，能自适应地调整搜索方向。
- 应用领域：
  - 组合优化
  - 机器学习
  - 信号处理
  - 自适应控制
  - 人工生命等领域。
- 是现代有关智能计算中的关键技术之一。

智能优化方法概述-模拟退火算法

- 1982年柯克帕特里克(Kirkpatrick)等将热力学中的退火思想引入组合优化领域，提出一种解大规模组合优化问题，特别是NP完全问题的有效近似算法。
- 模拟退火(Simulated Annealing,简称SA算法)。它*源于对固体退火过程的模拟*，采用Metropolis接受准则，并用一组称为冷却进度表的参数控制算法进程，使算法在多项式时间里给出一个近似最优解。
- 模拟退火法能以一定的概率接受差的能量，因而有可能跳出局部极小，但它的收敛速度较慢，很难用于实时动态调度环境

在网上搜索一个源程序，画出流程图，作为一个作业

智能优化方法概述-禁忌搜索算法

- 禁忌搜索法（tabu search）也是一种邻域搜索方法，从一个可行解 S 出发，S 产生邻域S'，如果 f 为目标函数，选取邻域中所有使 $f(S')$ 为最优的状态作为下一状态，并把这一移动的方向存入一个称为禁忌移动的表中
- 列在表中的移动在以后若干步内不允许再产生（*曾经走过的路不再重走*），直到它被从表中更新除去，这样可避免搜索进入循环状态。
- 每搜索一次，更新一次禁忌移动表，由于禁忌移动表的限制，有可能跳出局部极小。
- 用禁忌搜索法进行优化搜索时需要一可行的初始解。

在网上搜索一个源程序，画出流程图，作为一个作业

智能优化方法概述-蚁群算法

- 蚁群算法由意大利学者M. Dorigo, V. Maniez和A. Colorini等人在90年代首先提出
- 算法*受到真实蚁群觅食行为的启发*。虽然单个蚂蚁没有太多的智力，也无法掌握附近的地理信息，但整个蚁群却可以找到一条从巢穴到食物源之间的最优路线。
- 蚂蚁个体之间通过一种称之为信息素(pheromone) 的物质进行信息传递.
- 蚂蚁在运动过程中，能够在它所经过的路径上留下该种物质,而且蚂蚁在运动过程中能够感知这种物质,并以此指导自己的运动方向,因此,由大量蚂蚁组成的蚁群的集体行为便表现出一种信息正反馈现象:某一路径上单位时间走过的蚂蚁越多,表明该路线的可用性越好,则后来者选择该路径的概率就越大.蚂蚁个体之间就是通过这种信息的交流寻找最优的到达食物源的线路.
- 蚁群算法具有实现简单、正反馈、分布式的优点。

智能优化方法概述-粒子群算法

- 90年代中期，Eberhart博士和Kennedy博士共同发明了一种新的群体智能计算技术——粒子群优化算法(Particle Swarm Optimization，PSO)
- *源于对鸟群和鱼群群体捕食行为的研究*。
- PSO同遗传算法类似，是一种基于叠代的优化工具，是一种进化计算技术(evolutionary computation)。
- 粒子群优化算法的基本思想是**通过群体中个体之间的协作和信息共享来寻找最优解**。
- 系统初始化为一组随机解，通过叠代搜寻最优值。但是并没有遗传算法用的交叉(crossover)以及变异(mutation)，而是粒子在解空间追随最优的粒子进行搜索。
- PSO算法概念简单、容易实现、搜索速度快、搜索范围大，和其他优化算法相比，它的优点突出。
- 粒子群优化算法本质上是一种并行的全局性的随机搜索算法。

智能优化方法概述-人工神经网络

- 人工神经网络(Artificial Neural Networks),是*模仿或者模拟生物神经系统工作过程*，依照由大量简单神经元互联而构成的一种计算结构，并建立起来的一种数据处理技术，从而具备了解决实际问题的能力。
- 网络上的每个结点相当于一个神经元，可以记忆（存储）、处理一定的信息，并与其它结点并行工作。
- 求解一个问题是向人工神网络的某些结点输入信息，各结点处理后向其它结点输出，其它结点接受并处理后再输出，直到整个神经网工作完毕，输出最后结果。
- 特点和优越性
  - 具有自学习功能。
    - 预期未来的人工神经网络计算机将为人类提供经济预测、市场预测、效益预测，其前途是很远大的。
  - 具有联想存储功能。
  - 具有高速寻找优化解的能力。
    - 寻找一个复杂问题的优化解，往往需要很大的计算量，利用一个针对某问题而设计的反馈型人工神经网络，发挥计算机的高速运算能力，可能很快找到优化解。

在网上下载神经网络教材讲解

算法总结

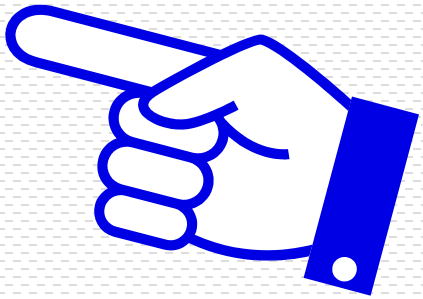
- 遗传算法
  - *借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）*
- 模拟退火算法
  - *源于对固体退火过程的模拟*
- 禁忌搜索算法
  - *曾经走过的路不再重走*
- 蚁群算法
  - *受到真实蚁群觅食行为的启发*
- 粒子群算法
  - *源于对鸟群和鱼群群体捕食行为的研究*
- 人工神经网络
  - *模仿或者模拟生物神经系统工作过程*
- ……
  - ……

经典问题

- 概述
- 约束优化问题（GA的基本概念）
  - 无约束优化
  - 非线性规划；
- 组合优化问题
  - 背包问题(knapasck)
  - 二次指派问题
  - 最小生成树问题
  - 旅行商问题
  - 影片递送问题
  - 集覆盖问题
  - 装箱问题
- 运输问题
  - 线性运输问题
  - 三维运输问题
  - 多目标运输问题
  - 固定费用运输问题
  - 工厂选址问题
- 网络设计与路径
  - 最短路径问题
  - 最大流问题
  - 最小成本流问题
  - 双准则网络设计问题
  - 多阶段工序计划问题
  - 集中式网络设计问题
  - 计算机网络扩展问题
- 设备布局设计问题
  - 单行机器布局问题
  - 多行机器布局问题
- 选址-分配问题
  - 一般选址-分配问题
  - 无能力约束的选址－分配问题
  - 障碍选址－分配问题
- 模糊优化（计划）
  - 模糊线性规划
  - 模糊非线性规划
- 调度问题
  - 流水车间调度问题
  - 作业车间调度问题
  - 机器调度问题

讲解思路

- 对每一个典型问题，力图从以下几个方面进行论述
  - 问题是什么
  - 如何根据问题构造数学模型
  - 模型的特点
  - 应用领域
  - 经典的求解方法（简介）



组合优化问题

(Combinatory Optimization Problems)

- 什么是组合优化问题
- 组合优化问题的特点
- 典型组合优化问题
- 组合优化问题的求解方法

什么是组合优化问题

- 定义：

$$\min \quad f(x)$$
$$s.t. \quad x \in S, \quad S \subset X$$
$$S, X: \text{拥有有限个或可数无限个解的离散集合}$$

- 假设是一个求极小的问题，目标函数为 *f(x)*,
- 问题的解 ***x*** 属于 ***S***，而 ***S*** 包含于 ***X***，***X*** 是解空间，***S*** 是可行解空间（可行区域）
- 所谓组合优化，是指在*离散的、有限的数学结构*上，寻找一个(或一组)满足给定约束条件并使其目标函数值达到最小的解。

组合优化问题的特点

$$\min \quad f(x)$$

$$s.t. \quad x \in S, \quad S \subset X$$

S, X：拥有有限个或可数无限个解的离散集合

- 如果 S 是有限集合的话，从理论上来讲，只要遍历所有的组合，就能找到最优解。
  - 然而，随着问题规模的增大，S 中解的个数会迅速增大，实际上要想遍历所有的解，几乎是不可能的。
- 一般来说，组合优化问题通常带有大量的局部极值点，往往是不可微的、不连续的、多维的、有约束条件的、高度非线性的NP完全(难)问题
- 组合最优化无法利用导数信息
- 精确地求解组合优化问题的全局最优解的“有效”算法一般是不存在的。

组合优化的研究

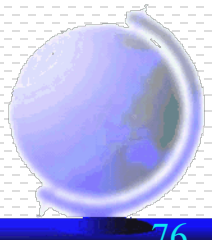
- 怎么才能把一些社会现象、活动等捕捉归纳为组合优化问题？
- 各种组合优化问题具有什么特征，拥有什么性质？
- 怎么才能够在尽可能短的时间内求解组合优化问题？
- 为了构造快速解法，什么样的性质是有用的？

典型组合优化问题

- 在日常生活中，特别是在工程设计中，有许多组合优化问题。
- 其中一个重要而广阔的应用领域就是如何有效地利用不充足的资源以提高生产效益。典型问题有：
  - 集覆盖问题(set-covering problem)
  - 装箱问题(bin-packing problem)
  - 背包问题(knapsack problem)
  - 指派问题(assignment problem)
  - 旅行商问题(traveling salesman problem)
  - 影片递送问题(film delivery problem)
  - 最小生成树问题(minimum span tree problem)
  - 作业调度问题(job-shop scheduling problem)
  - 车辆路径与调度问题（Vehicle Routing Problem)

组合优化问题求解方法分类

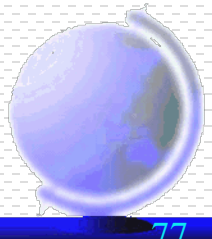
- 从求解方法的大的分类上，可以分为：
- 精确的求解方法
  - 可以得到问题的最优解，对小规模问题适用，当问题的规模较大时，一般无法在有限的时间内获得问题的最优解
  - 对于组合优化问题，通常采用的是近似求解方法。
- 近似的求解方法，可以进一步划分为：
  - 以确保解的精度为目标的方法
  - 以尽可能获得更好解为目标的方法，包括：
    - 启发式(Heuristics)
    - 亚启发式(Meta-Heuristics)



组合优化问题求解方法分类--近似求解方法

- 以确保解的精度为目标的方法
  - 能保证一定的精度，且成本较低，例如：
    - 程序所获得的目标函数值和最优解的目标函数值相比，达到90%或99%以上，而且获得这样的解的成本不超过获得最优解的成本的10%或20%，这样的算法是可接受的。
  - 当然，从数学上准确地保证并不是一件简单的事情
  - 这类近似方法的研究，会产生很多有趣的数学特征，吸引了很多从事理论研究的学者，从事这方面的工作。

如何评价算法的好坏



组合优化问题求解方法分类--近似求解方法

- 以尽可能获得更好解为目标的方法

  这类方法侧重于实际应用，有很多方法都是从实用的角度出发来考虑问题的。

  - 启发式(Heuristics)方法
    - 即使我们求解很多的应用问题，也不会产生精度很差的解，偶尔，对于非常棘手的问题也许会产生精度很差的解，但一般的场合下，应该没有问题。
  - 亚启发式(Meta-Heuristics)方法
    - 启发式方法中，一种被称之为亚启发式的方法，得到了广泛的研究，发现了一些较好的求解方法
    - GA就是其中之一，另外还有 TS，SA，PSO等算法。

组合优化问题求解方法分类--近似求解方法

- 亚启发式(Meta-Heuristics)
  - 从算法的角度来讲，是指不依赖于特定问题的启发式算法。
  - 其算法的基本框架被设计成不论对什么样的问题都具有通用性。
  - 一般情况下，亚启发式算法要比特定问题专用的启发式算法的解的精度要差。
  - *所以，针对特定的问题，要精心设计特定的算法。*
    也有人把以邻域搜索为基础的组合优化方法统称为亚启发式

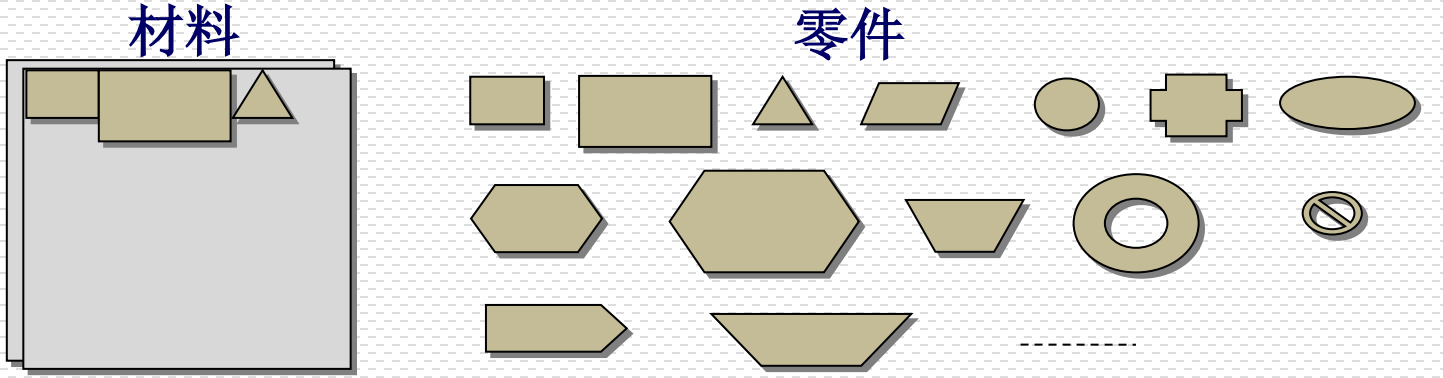
组合优化问题的典型求解方法

- 求解方法有很多，以下列举几种典型的方法：
  - 完全枚举法
  - 准完全枚举法
  - 降序排列法
  - 贪婪法
  - 随机法
  - 松弛法
  - 分割法
  - 分支定界法
  - 邻域搜索法
  - 多起点邻域搜索法
  - 人工智能法

从概念上来讲，这里所列的一些方法，有些是具有涵盖关系，比如说降序排列法，也应该属于贪婪法的一种。

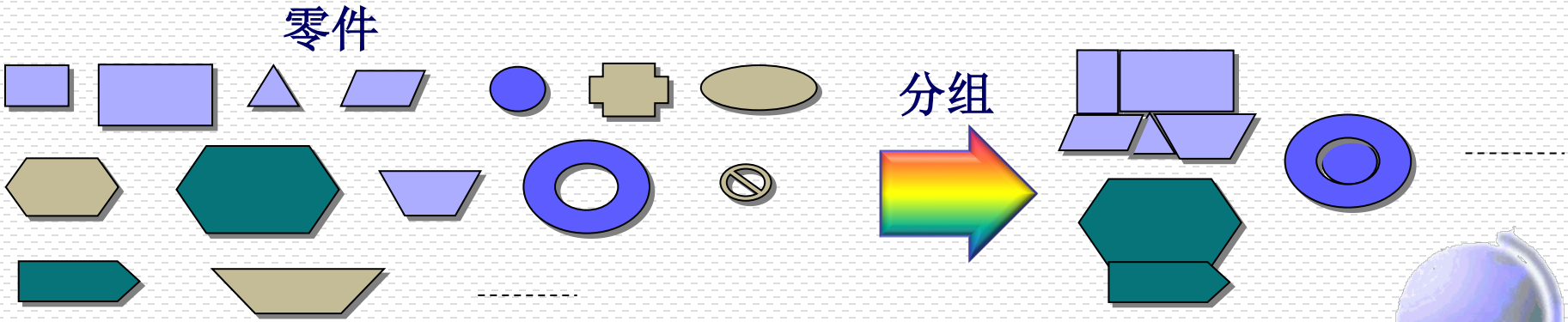
典型求解方法--完全枚举法

- 枚举所有的组合，从中选择最好的组合的方法
- 这是一种最简单的方法，可以找到问题的最优解
- 也是一种精确的解法
- 但随着问题规模的扩大，会导致组合爆炸，使得完全枚举变得不可能
  - 例如：考虑把材料分配给 *n* 个零件的问题（即零件搭配组成图案）。我们可以按顺序给零件分配材料，一块材料无法满足要求的话，就分配下一块材料。这样我们可以得到一种分配结果。
  - 总的分配结果数，也就是零件的组合数，等于 *n* 个零件的排列，*n* 的阶乘。



典型求解方法—准完全枚举法

- 通过分组，使组合数降低，使完全枚举成为可能的一种方法
  - 正如在完全枚举法中所举例的那样，随着零件数量的增加，全部枚举零件的组合非常困难。
  - 然而如果按零件的尺寸、形状的特征进行分组的话，组的组合就会减少，其结果导致有可能采用枚举的方法进行求解。
  - 尤其是，当把多个零件组合成一个零件进行处理时，零件的数量、组合的数量、计算时间等都会显著减少。
- 未必是问题的最优解



典型求解方法—降序排列法

- 把零件按尺寸大小顺序排列，并分配材料，以此结果作为最优组合的方法。
- 想法很单纯，材料分配率也不坏。因为只进行一次分配，所以几乎不花费计算时间。
- 最简单的方法之一，未必能得到最优解

典型求解方法--贪婪算法（greedy method）

- 采用逐步构造最优解的方法。在每个阶段，都作出一个看上去最优的决策（在一定的标准下）。决策一旦作出，就不可再更改。
- 作出贪婪决策的依据称为贪婪准则（**greedy criterion**）。
- 一种近似求解方法
- 货箱装船、机器调度、最短路径、背包问题等方面都有应用

典型求解方法--贪婪算法（greedy method）

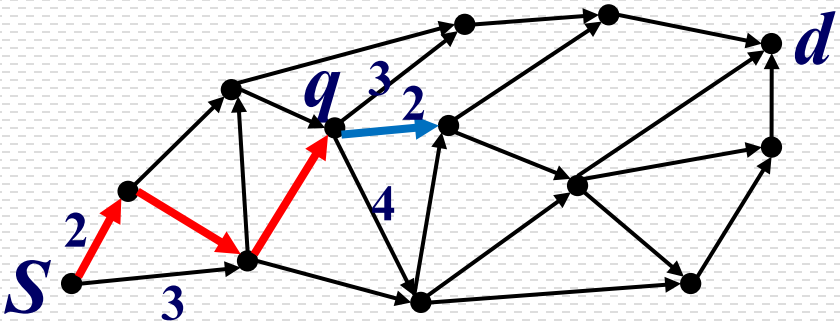
- 例 [找零钱]：
  - 一个小孩买了价值少于1美元的糖，并将1美元交给售货员。售货员希望用数目最少的硬币找给小孩。
  - 假设硬币数目不限，面值为2 5、1 0、5、及1美分。
  - 售货员分步骤组成要找的零钱数，每次加入一个硬币。选择硬币时所采用的贪婪准则如下：
    - 每次都选择面值尽可能大的硬币，使找完的零钱数额尽量增大
    - 最终的零钱数额等于要找的零钱数额
  - 假设需要找给小孩67美分，

25*2 =50美分 + 10=60美分 + 5=65美分 + 1*2=67美分

- 贪婪算法有种直觉的倾向，在找零钱时，直觉告诉我们应使找出的硬币数目最少（至少是接近最少的数目）。可以证明采用上述贪婪算法找零钱时所用的硬币数目的确最少。

典型求解方法--贪婪算法（greedy method）

- 例 [最短路径]：
  - 给出一个有向网络，要求找一条从初始点 *s* 到达目的点 *d* 的最短路径。
  - 贪婪算法分步构造这条路径，每一步加入一个顶点。
  - 假设当前路径已到达点*q*，且顶点*q* 并不是目的点*d*。
  - 加入下一个顶点所采用的贪婪准则为：
    - 选择离*q* 最近且目前不在路径中的顶点。



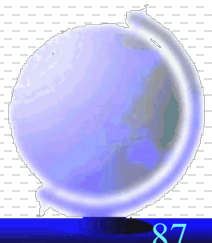
- 这种贪婪算法并不一定能获得最短路径。

典型求解算法--随机法

- 随机选择一些解，在所发现的解中选择目标函数值最小的解的方法

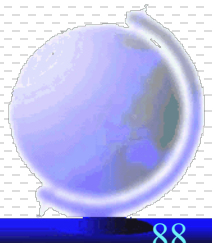
典型求解算法——松弛法

- 通过松弛约束条件，获得问题的一个解，然后对其进行修正，从而构成可行解的一种方法。
- 一种近似求解方法



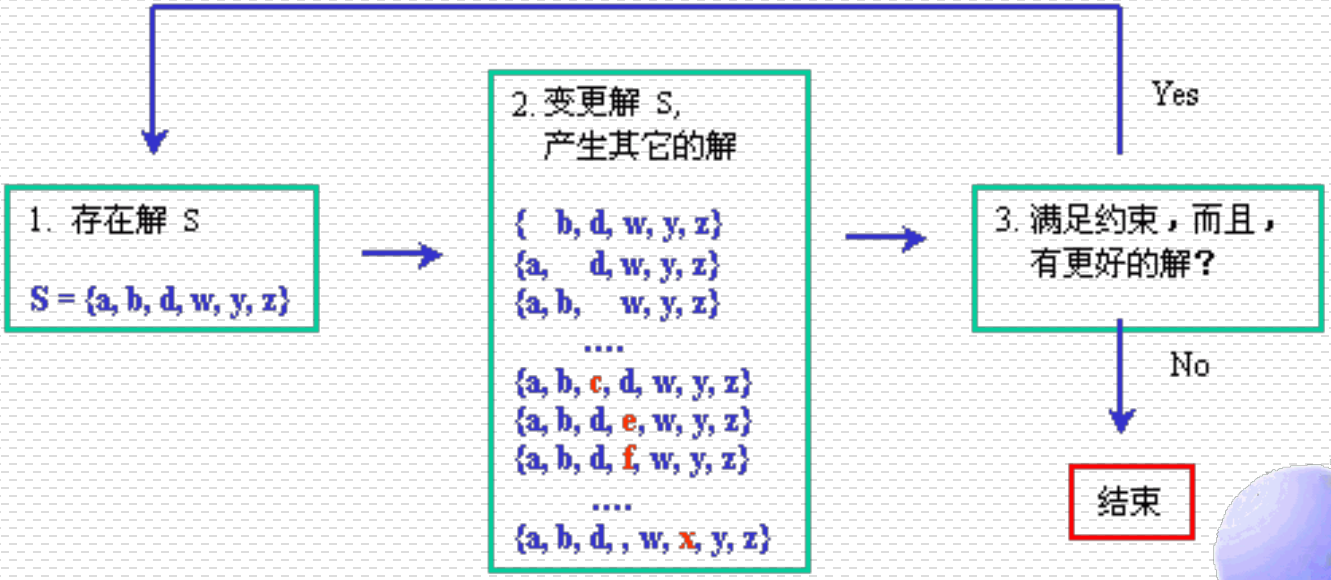
典型求解算法—分割法

- 将给定的问题分解为几部分，从各部分的解中合成原问题的可能解的一种方法。
- 近似求解方法
- 例如对于TSP问题：
  - 可以把给定的城市数分为多个组，各组内的最优路径找到以后，合成全体的最优路径，就相当于这种方法。



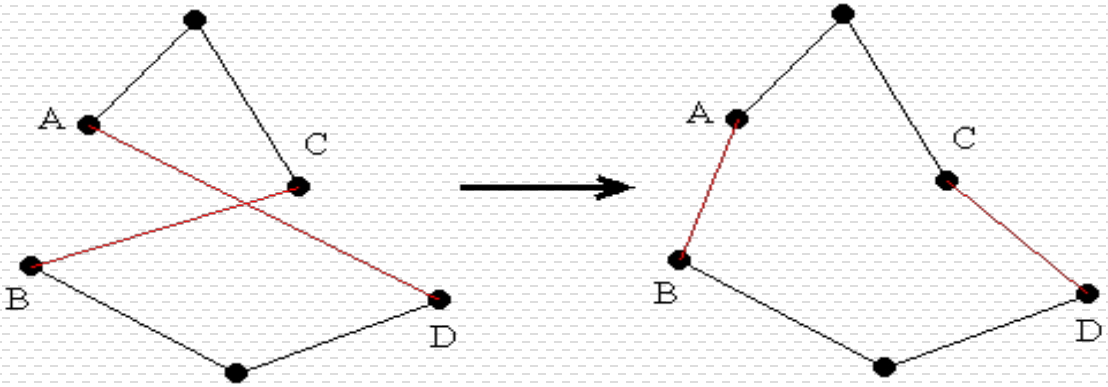
典型求解算法--邻域搜索法

- 也称为：局部搜索法、逐次改善法
- 亚启发式的最基本的方法
- 首先找到问题的一个解，为了改善解的目标函数值，对解做一些变更，如果通过解的变更使目标值改善的话，就继续循环，否则终止搜索、输出解。
- 局部搜索未必能得到问题的最优解，有可能陷入局部最优。



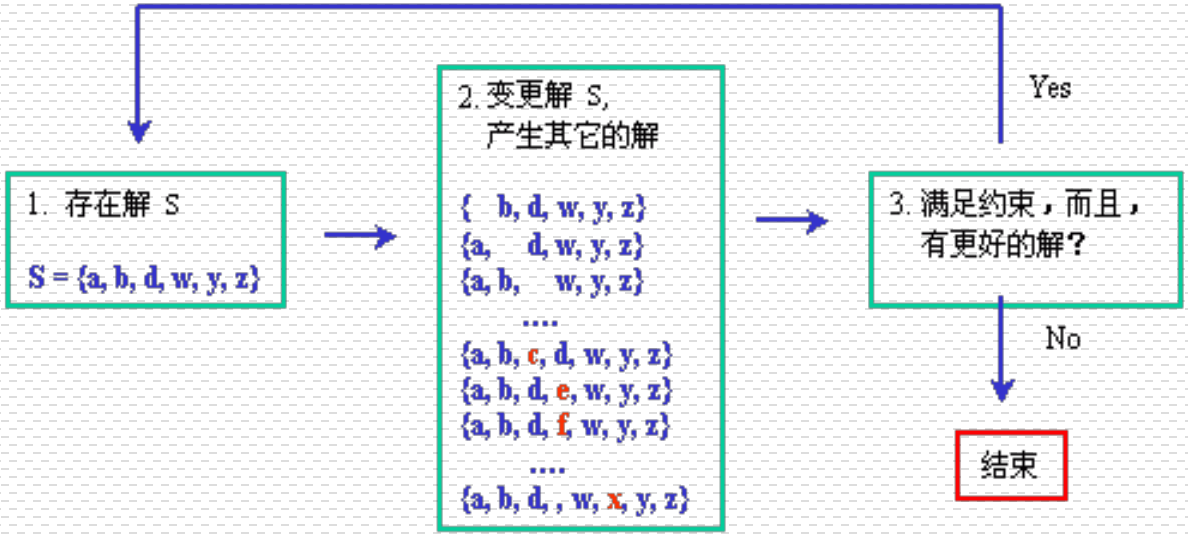
典型求解算法--邻域搜索法

- 例如：在TSP问题当中，首先我们可以找到一个回路，然后我们可以在这个回路的基础上，交换一些分支这样会得到新的回路，如果得到改善，就替换当前的解，直至无法改善为止。



- 局部搜索方法未必能得到问题的最优解，有可能陷入局部最优。

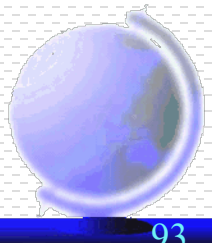
典型求解算法--邻域搜索法



- 如何才能改善解的精度，需要*下些功夫、想些办法、找些技巧*。
- 例如可以考虑增加2个减少2个、1个增加1个减少等等多种方法。当然变化越多花费的时间就越多，但未必就能使解的精度改善得更好。

典型求解算法—多起点邻域搜索法

- 基本想法：
  - 起点不同，所获得的解也不同，进行多次搜索的话，可以从中找到更好的解。
- 想法简单，据报道是非常有效的。

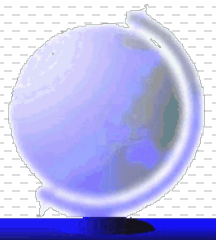


典型求解算法——人工智能算法

- 神经网络
- 遗传算法
- ……

- 由于要进行高度复杂的计算，所以计算效果比较好，当然也需要花费更长的计算时间。

机器学习算法



机器学习核心

1 机器学习核心

2 机器学习发展史

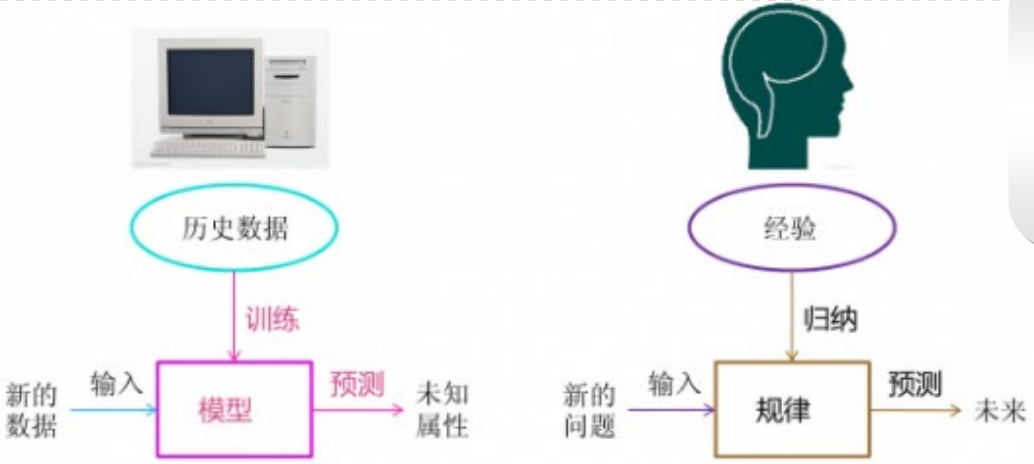
3 常用机器学习算法

4 未来发展趋势

机器学习的概念与原理

- 机器学习是能够赋予机器一种学习能力，能够完成直接编程无法完成的任务。机器学习通过数据，学习到数据中的特点，训练出模型，然后利用模型来预测未来情况。

- 从以“推理”为重点到以“知识”为重点，再到以“学习”为重点
- 机器可以自动“学习”的算法，即从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法。目前，机器学习=“分类”



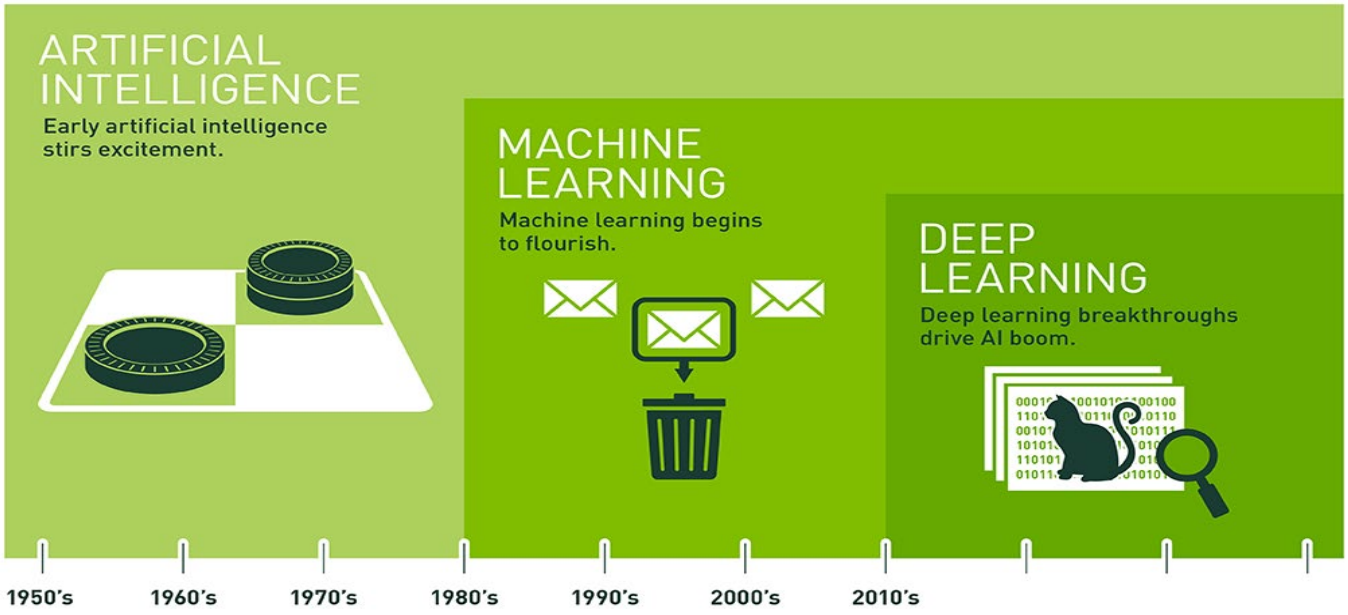
$x \rightarrow F(x) \rightarrow y$

- Class label (Classification)
- Vector (Estimation)

机器学习的核心点

- 数据是机器学习的来源；算法是机器学习的逻辑
- 相关是支撑机器学习的核心概念
- 机器学习的目标：学习的模型能够很好的适用于“新样本”

■ 机器学习、深度学习和人工智能之间的关系



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

机器学习发展史

机器学习发展史

1943-1969 第一代神经网络

人工神经网络
感知机
XOR问题

1986-1998 第二代神经网络

反向传播
多层感知机
梯度消失

1986-2006 统计学习的春天

SVM
朴素贝叶斯
聚类算法

2006-2012 深度学习的到来

Relu函数
深度置信网络
不用预处理

2012-2016 深度学习的发展

CNN
RNN
GAN
LSTM

2016至今 深度学习的未来

AlphaZero
Capsule
深度强化学习
迁移学习

常见的机器学习算法

强化学习	非监督学习	监督学习
	神经网络	线性回归
	K-means	逻辑回归
	PCA	SVM
		神经网络
		朴素贝叶斯
		KNN

Supervised Learning :

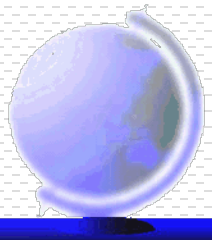
- **given data，predict labels**

Unsupervised Learning :

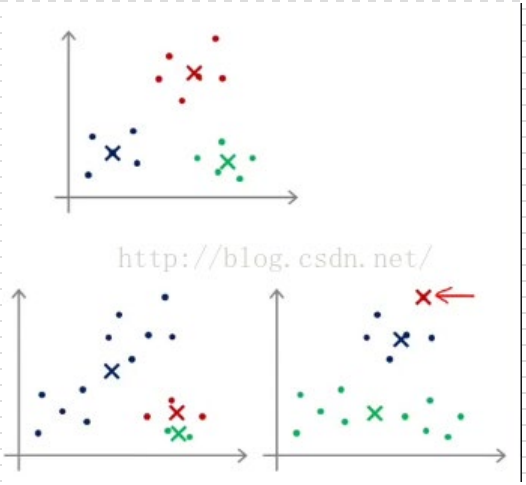
- **given data ,learn about that data**

Reinforcement Learning :

- **given data , choose action to maximize expected long-term reward**

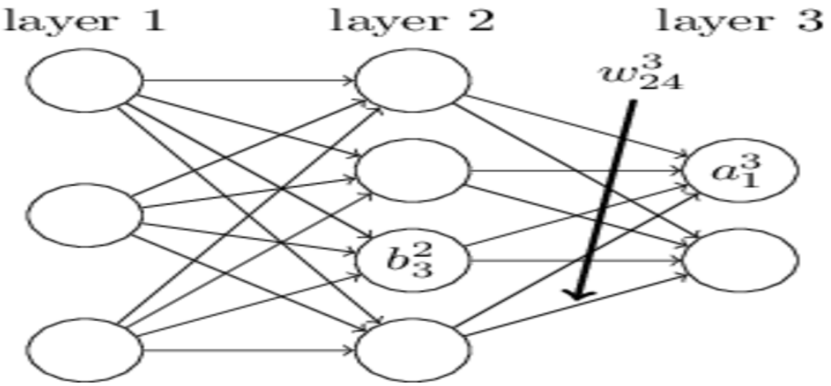


K-means



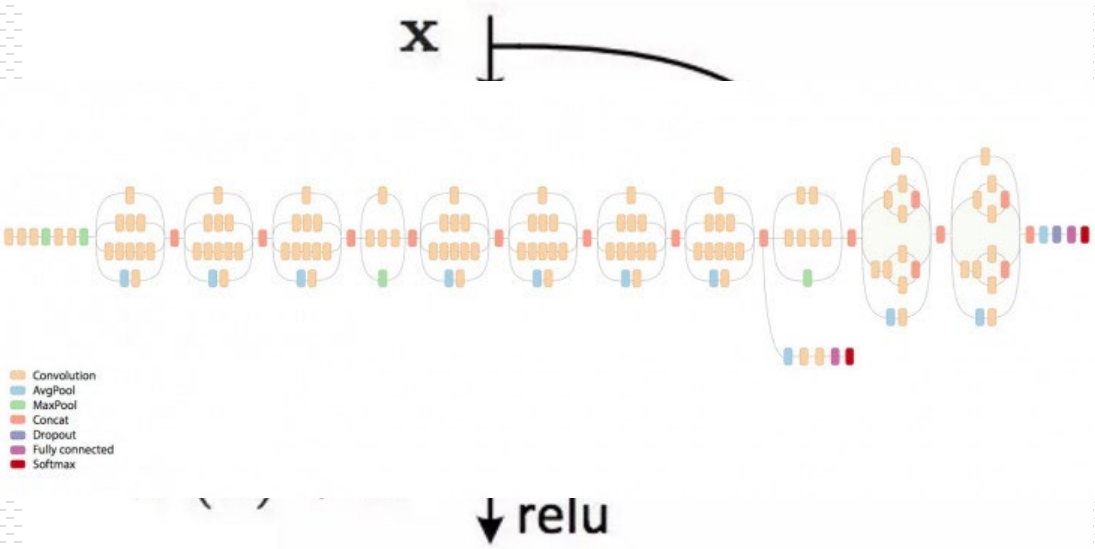
1、首先选择K个随机的点，称为聚类中心（cluster centroids）；
2、对于数据集中的每一个数据，按照距离K个中心点的距离，将其与距离最近的中心点关联起来，与同一个中心点关联的所有点聚成一类。
3、计算每一个组的平均值，将该组所关联的中心点移动到平均值的位置。
4、重复步骤2-4直至中心点不再变化。

神经网络



这是一个包含三个层次的神经网络。layer 1的是输入层，layer 3的是输出层，layer 2的是中间层（也叫隐藏层）。输入层有3个输入单元，隐藏层有4个单元，输出层有2个单元。

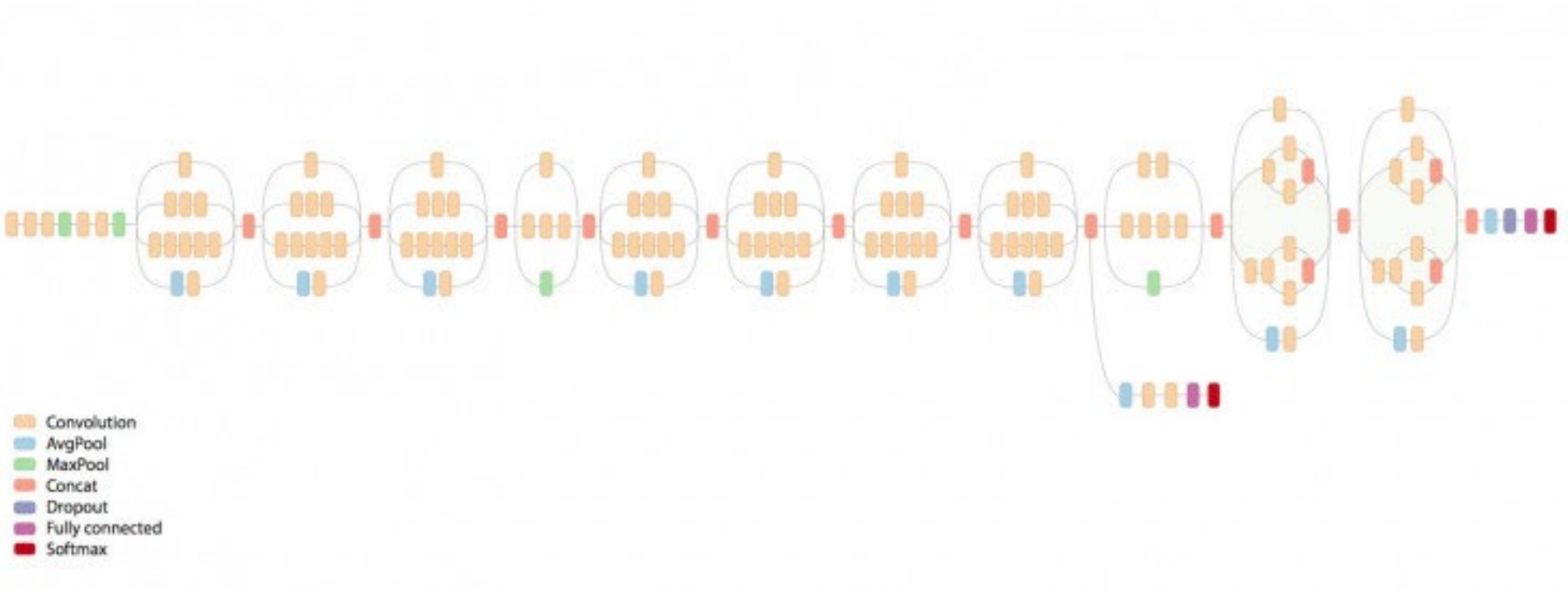
深度学习-ResNet



ResNet有效的解决了深度卷积神经网络难训练的问题。这是因为在误差反传的过程中，梯度通常变得越来越小，从而权重的更新量也变小。

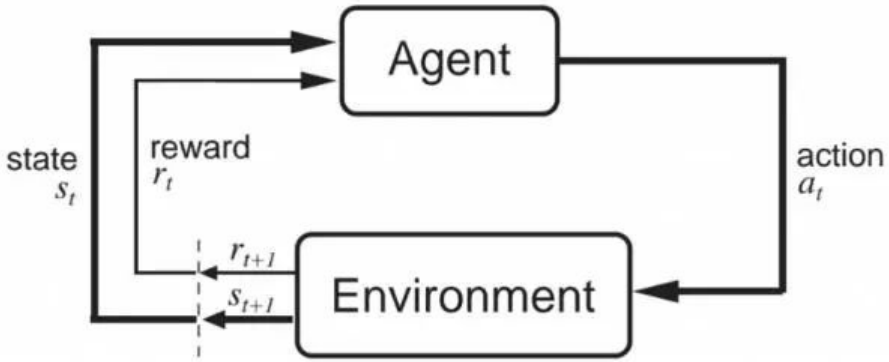
ResNet通过增加跨层的连接来解决梯度逐层回传时变小的问题。

深度学习-ResNet



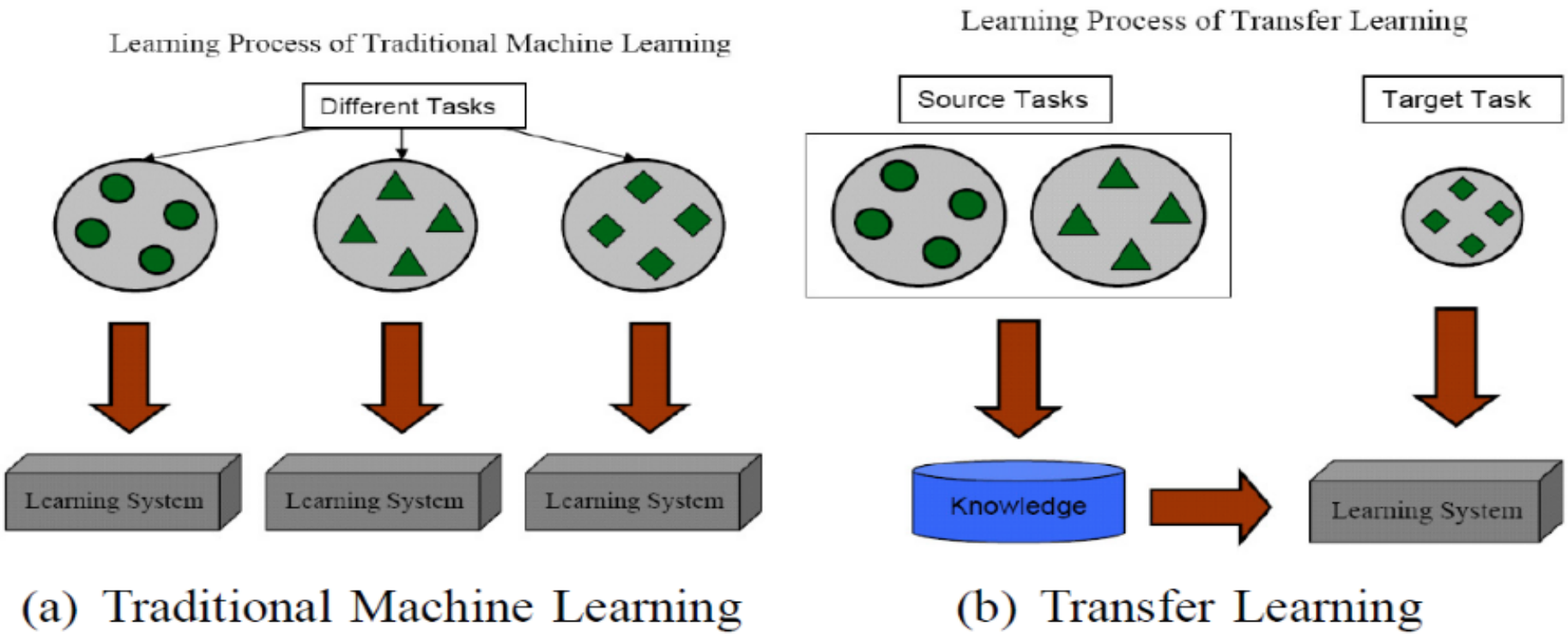
2015年ImagetNet的第一名使用的简单图例

强化学习-ResNet



- environment（标准的为静态stationary，对应的non-stationary）
- agent（与环境交互的对象）
- 输入：
- 状态（States）= 环境，例如迷宫中每一个格都是一个state
- 动作（Actions）= 每个状态下，有什么行动是容许的
- 奖励（Rewards）= 进入每个状态时，能带来的正面或者负面的价值（utility）
- 输出：
- 方案（Policy）= 每个状态下，你会选择哪个行动

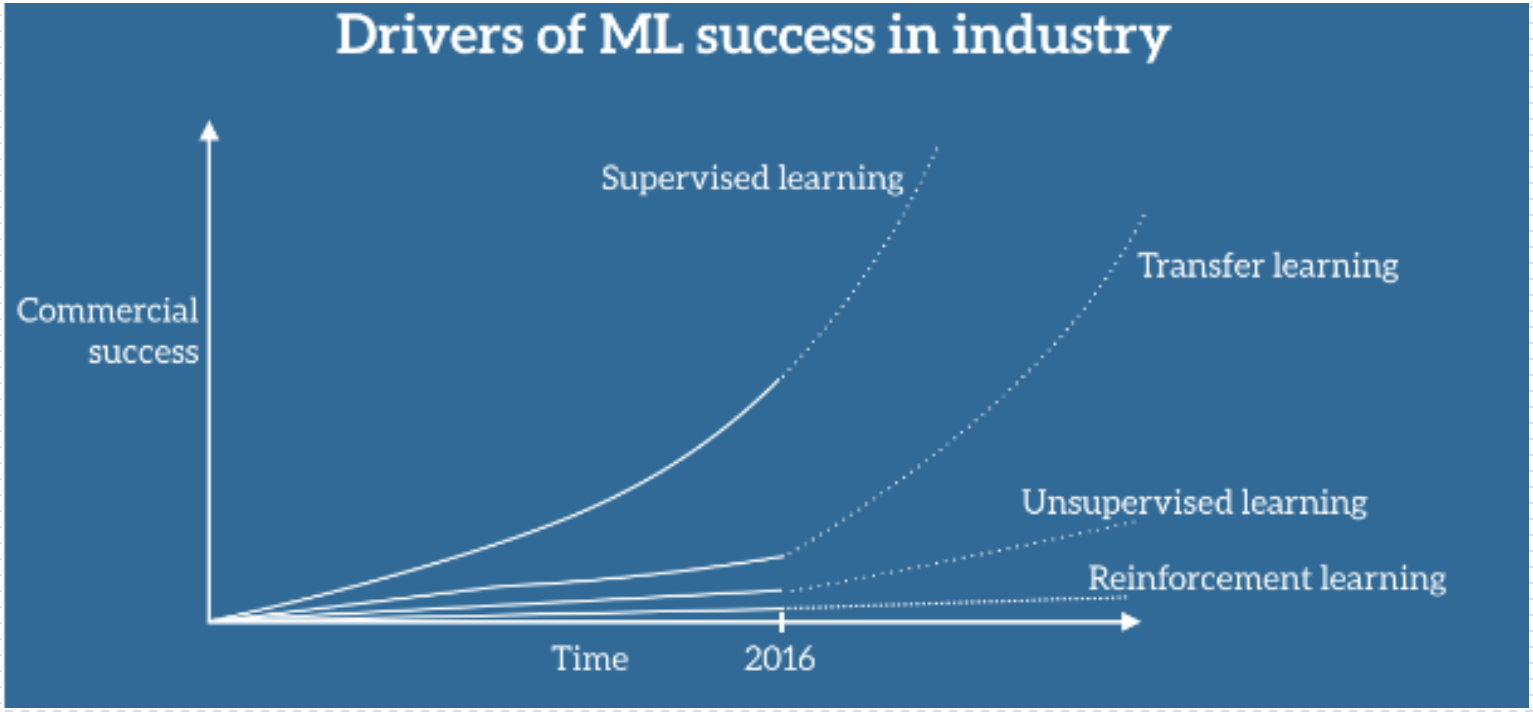
迁移学习



(a) Traditional Machine Learning (b) Transfer Learning

运用已有的知识来学习新的知识，核心是找到已有知识和新知识之间的相似性，用成语来说就是举一反三。

未来发展趋势


Drivers of ML success in industry
Supervised learning
Transfer learning
Commercial success
Unsupervised learning
Reinforcement learning
Time
2016

End