

## **A**

- A2C (advantage actor-critic) 472
- A2C (advantage actor-critic) algorithm
  - balancing CartPole using 473
- A2C (advantage actor-critic) architecture 459
- AA (alliance algorithm) 60
- A (adjacency matrix) 410
- A\* algorithm 105
  - for routing 149
- abc\_algorithm function 390
- ABC (artificial bee colony) 59, 235, 322, 324, 362
- ABC (artificial bee colony) algorithm 384, 385
- acceleration coefficients 332
- acceptance probability 165–167
- Ackley function 178
- ACO (ant colony optimization) 59, 235, 266, 322,  
324, 362, 365–369, 393
  - simple ACO 370
  - solving TSP with 376
  - variants of 369
- acopy.Colony object 381
- acopy library 380
- acopy solver 382
- ACS (ant colony system) algorithm 374
- action-value function 456
- Actor class 474
- actor-critic methods 459
- acyclic graph 66
- Adam optimizer 431, 475
- adaptation in SA 172–175
- adaptive black hole algorithm 60
- adaptive GA 298
- adaptive memory 197
- adaptive tuning 206
- addition 343
- adjacency-based problems 283
- adjacency list 286
- age-based selection 260
- AI (artificial intelligence) 243, 398
  - AI-empowered daily routines 398
- ALBP (assembly line balancing  
problem) 221–230
- algorithms
  - genetic algorithms
    - solving traveling salesman problem 300–304
  - nature-inspired algorithms 59
  - search algorithms 20
  - unsupervised machine learning 438–441
- alleles 241
- alliance algorithm (AA) 60
- amortized optimization 427–432
- ANN (artificial neural network) 422, 449
- annealing process 168
- ant colony optimization 363–366
- ant colony optimization (ACO) 59, 324
- ant colony system (ACS) algorithm 374
- ant cycle model 369
- ant density model 368

ant quantity model 368  
 ant system (AS) algorithm 373  
 ant\_tour function 377  
 A-PSO (asynchronous PSO) 331  
 argmax 359  
 artificial bee colony (ABC) 59, 324  
 artificial bee colony (ABC) algorithm 385  
 artificial bee colony algorithm 383–385  
 artificial intelligence (AI) 243  
 artificial neural network (ANN) 422  
 ASA-GS (adaptive simulated annealing with greedy search) 174  
 A\* search algorithm  
     CH query phase 135  
 A\* search algorithm 124–131  
     CH example 136–144  
     CH preprocessing phase 131–135  
 ASF (augmented scalarization function) 296  
 aspiration criteria 205  
 aspiration criterion 198  
 assembly line balancing problem 221–230  
 Atkinson–Shiffrin model 201  
 ATSP (asymmetric TSP) 27  
 attention mechanisms 416  
     overview 416  
 augmented graph 131  
 autocatalytic behavior 366  
 autonomous coordination in mobile networks  
     using PPO 481–485  
 AverageMeter class 443, 446

## B

BA (bat algorithm) 59, 235, 324  
 backpropagation 353  
 backtracking algorithm 182  
 bacterial foraging optimization algorithm  
     (BFO) 60, 324  
 bacterial swarming algorithm (BSA) 60, 324  
 balancing  
     CartPole using A2C and PPO 473  
 banana function 387  
 basic units 45  
 bat algorithm (BA) 59, 324  
 batch\_data 444  
 batch\_labels 444  
 batch\_lengths 444  
 batch size 435  
 batch\_size 443

batch\_size parameter 435  
 BBBC (big-bang big-crunch) 60  
 BBO (biogeography-based optimization) 60  
 beam search 105  
     for routing 148  
 beam search algorithm 121–124  
 best\_distance variable 382  
 best\_path variable 382  
 BFO (bacterial foraging optimization algorithm) 60, 324  
 BFS (breadth-first search) 73  
 BFS (breadth-first search) 74–83  
 BGA (binary-coded GA) 250  
 BGA (binary genetic algorithm) function 264  
 BH (black hole algorithm) 60  
 BHMO (black hole mechanics optimization) 60  
 bias term 351  
 bidirectional Dijkstra search 96  
 big-bang big-crunch (BBBC) 60  
 big O notation 80, 81, 82  
 bill\_depth\_mm variable 354  
 bill\_length\_mm variable 354, 360  
 binary bridge experiment 365  
 binary PSO 340–342  
 bin() function 262  
 biogeography-based optimization (BBO) 60  
 bird flocking 326  
 black hole algorithm (BH) 60  
 black hole mechanics optimization (BHMO) 60  
 blind search algorithms 62, 104  
     graph search 72  
     shortest path algorithms  
         bidirectional Dijkstra search 96  
         UCS 94  
 blind search algorithms  
     applying to routing problem 98–101  
     graphs 63–72  
     graph traversal algorithms 74–89  
         breadth-first search 74–83  
         depth-first search 84–89  
     shortest path algorithms 89–97  
         Dijkstra's search 90–94  
 BMU (best matching unit) 422  
 BOA (butterfly optimization algorithm) 59  
 body\_mass\_g variable 354  
 Bohachevsky function 178  
 Borůvka's algorithm 105  
 boundary mutation 278

BPSO (binary PSO) 340  
 brainstorm optimization algorithm (BSO) 60  
 bridges with non-equal lengths experiment 366  
 BSA (bacterial swarming algorithm) 60, 324  
 BS (bidirectional search) 74  
 BSO (brainstorm optimization algorithm) 60  
 BSs (base stations) 481  
 Bukin function 178  
 butterfly optimization algorithm (BOA) 59

## C

CA (cultural algorithm) 245  
 CA (cultural algorithms) 59, 234  
 can\_attack function 35  
 CartPole  
   balancing using A2C and PPO 473  
 cat swarm optimization (CSO) 59  
 Cauchy distribution 278  
 Cauchy mutation 278  
 c-convex class 429  
 c\_convex functions 431, 432  
 c\_convex objects 430  
 celestial coordinate system 428  
 celestial\_to\_euclidean() function 427  
 CH (contraction hierarchies) 105, 154  
 CH (contraction hierarchies) algorithm  
   query phase 135  
 CH (contraction hierarchies) algorithm  
   example 136–144  
   preprocessing phase 131–135  
 CH (contraction hierarchy) 105  
 chromosome\_length argument 262  
 chromosomes 241  
 CI (computational intelligence) 243  
 cities dictionary 376  
 classification 402  
 closed-loop system 305  
 CLT (central limit theorem) 235  
 clustering 402  
 CMAB (contextual multi-armed bandit) 452, 470, 486  
 CNNs (convolutional neural networks) 402, 409  
 COA (cuckoo optimization algorithm) 59  
 CoE (co-evolution) 59, 234, 245  
 co-evolution (CoE) 59  
 cognitive acceleration coefficient 349  
 cognitive component 329, 346  
 cognitive parameter 359  
 COIN-OR branch and cut (Cbc) 184  
 combinations 26  
 combined criteria 307  
 competitive learning 422  
 computational intelligence (CI) 243  
 constrained depth-first search 89  
 constraints 14, 37–45  
 constraint-satisfaction problems (CSPs) 13  
 constraint satisfaction problems (CSPs)  
   solving with tabu search 207–213  
 context 470  
 contextual bandits  
   truck selection problem, solving using 485–490  
 continuous optimization  
   function optimization 175–180  
 continuous problems  
   solving 213  
 continuous PSO 326  
   algorithm 326  
 contraction hierarchies  
   for routing 151–153  
 control problems 48  
 conv1 layer 413  
 conv2 layer 413  
 convex hull  
   finding 441  
 ConvexHull function 447  
 ConvexNet model 443, 444, 447  
 convolution operation 409  
 cooling\_alpha parameter 179  
 cooling\_schedule parameter 179  
 Cora dataset 412  
 cost function 10  
 cost\_function 376  
 cost property 382  
 criterion function 10  
 Critic class 475  
 Crook's pencil-and-paper algorithm 182  
 cross-entropy loss 352  
 cross-in-tray function 178  
 crossover function 264  
 crossover methods  
   permutation-based genetic algorithms  
     cycle crossover 288  
     edge crossover 286  
     order 1 crossover 287  
     partially mapped crossover 283

crossover methods  
     permutation-based genetic algorithms 283–289  
 crossover operator 244  
 crossover parameter 281  
 crossover\_prob argument 264  
 crossover rate 318  
 CSA (classical simulated annealing) 164  
 CS (cuckoo search) 59  
 CSO (cat swarm optimization) 59  
 CSP (constraint-satisfaction problem) 181  
 CSPs (constraint-satisfaction problems) 13, 33  
 CSPs (constraint satisfaction problems)  
     solving with tabu search 207–213  
 cuckoo optimization algorithm (COA) 59  
 cuckoo search (CS) 59  
 CUDA\_DEVICE\_ORDER environment  
     variable 434  
 CUDA\_VISIBLE\_DEVICES environment  
     variable 434  
 cultural algorithms (CA) 59  
 CurveFittingProblem class 281  
 CVRP (capacitated vehicle routing problem) 29,  
     471  
 cycle 66, 288  
 cycle crossover 288

## D

DA (dragonfly algorithm) 59  
 Data class 442  
 DataParallel 436  
 dataset object 436  
 dataset path 435  
 data.train\_mask 414  
 DBSCAN (density-based spatial clustering) 402  
 DDPG (deep deterministic policy gradient) 472  
 DEAP (Distributed Evolutionary Algorithms in  
     Python) 266  
 debug parameter 179  
 decision science 9  
 decision variables 9  
 decision variables  
     number and type of 26–32  
 DE (differential evolution) 59, 234, 245, 266  
 DeepFreight 471  
 DeepPool 471  
 DefaultRoutingSearchParameters() method 217  
 delta\_pheromones matrix 378  
 derivative term 305

deterministic algorithms 50  
 deterministic tuning 205  
 device 435  
 device variable 443  
 DFS (depth-first search) 73  
 DFS (depth-first search) 84–89  
 differential evolution (DE) 59  
 dijkstra\_path function 100  
 Dijkstra's search 90–94  
 dimensions 349  
 directed graph 64  
 Disp class 442  
 Disp\_results helper function 445  
 distance\_callback function 216, 217  
 distance\_matrix 215  
 distance\_matrix dictionary 376  
 districts 45  
 diversification 20  
 DL-assisted heuristic tree search (DLTS) 426  
 DL (deep learning) 403  
 DLS (depth-limited search) 73  
 DLX (Dancing links) 182  
 DNA (deoxyribonucleic acid) 241  
 DNNs (deep neural networks) 426  
 dolphin partner optimization (DPO) 59  
 dolphin swarm optimization algorithm  
     (DSOA) 59  
 Domingos, Pedro 399, 400  
 downward graph 135  
 DP (dynamic programming) 28  
 DPO (dolphin partner optimization) 59  
 DQN (deep Q network) 472  
 dragonfly algorithm (DA) 59  
 drop wave function 178  
 DSOA (dolphin swarm optimization  
     algorithm) 59  
 dual\_annealing() function 176  
 dynamic programming algorithms 19  
 dynamic tabu tenure 204

## E

EasyGA 267  
 EC (evolutionary computation) 243  
 EC (evolutionary computation) algorithms 234  
 ED (edge difference) 151  
 edge crossover 286  
 edge difference (ED) 133  
 edges 404

edge table 286  
 Eggholder function 178  
 ElementwiseProblem class 302  
 eliminate\_duplicates parameter 281, 295  
 elitism 252  
 elitist selection 261  
 employed bees 386  
 ENC(v) encoder 407  
 end-to-end learning 425  
 enumerate function 487  
 EPA (Environmental Protection Agency) 12  
 EP (evolutionary programming) 59, 234, 245  
 episode\_rewards list 477  
 epsilon\_greedy(epsilon) function 468  
 epsilon-greedy strategy 466  
 error calculation 353  
 escape mechanism 207  
 ES (evolutionary strategies) 59, 234, 245  
 estimated time of arrival (ETA) 404  
 ETA (estimated time of arrival) 404  
 eta parameter 295  
 euclidean\_to\_celestial() function 427  
 eureka 53  
 evaluation criteria 16  
 evolutionary computation
 

- biology fundamentals 241
- theory of evolution 242

 evolutionary computation 241–246
 

- evolutionary computation 243–246

 evolutionary programming (EP) 59  
 evolutionary strategies (ES) 59  
 expectation operator 463  
 exploitation 20  
 exploitation search dilemma 22  
 exploit\_only\_greedy() function 468  
 exploit-only greedy strategy 466  
 exploration 20  
 exploration search dilemma 22  
 explore\_only() function 468  
 explore-only strategy 466  
 exponential cooling schedule 171

## F

FA (firefly algorithm) 59, 235, 324  
 far-sighted 456  
 feasible solution 199  
 feasible solutions 7  
 feasible states 16

feedforward method 353  
 F field 280  
 FIFO (first in, first out) 75  
 filename parameter 435  
 final temperature 172  
 final\_temp parameter 178  
 find\_cycle() method 111  
 finding convex hull 441  
 finite catenary problem 44  
 firefly algorithm (FA) 59, 324  
 fish school search (FSS) 59  
 fit function 55  
 fitness function 244, 249  
 fitness-proportionate selection 253  
 fitness\_score function 262  
 fitness update 330  
 flipper\_length\_mm variable 354, 360  
 FloatRandomSampling 310  
 FloatRandomSampling class 268, 269  
 FloatRandomSampling operator 281  
 flower pollination algorithm (FPA) 60  
 FLP (facility location problem) 42  
 FNNs (feedforward neural networks) 402  
 FOA (forest optimization algorithm) 60  
 folium library 71  
 forest optimization algorithm (FOA) 60  
 forgetting problem 416  
 forward function 413  
 forward method 474  
 forward\_prop function 357  
 forward(xyz) method 429  
 FPA (flower pollination algorithm) 60  
 FPS (fitness-proportionate selection) 253, 261  
 frequency-based memory 201  
 FSA (fast simulated annealing) 164  
 FSS (fish school search) 59  
 function optimization 175–180
 

- solving using machine learning 427–432
- solving using supervised machine learning 427–432

 Furuta Pendulum Robot 472

## G

G0 point 44  
 GA class 281  
 GA (genetic algorithm) 59, 234, 245
 

- building blocks 246
- fitness function 249

- reproduction operators
  - crossover 257
  - mutation 259
  - new population 259
- selection operators
  - elitism 252
  - fitness-proportionate selection 253
  - random selection 257
  - rank-based selection 254
  - stochastic universal sampling 255
  - tournament selection 256
  - survivor selection 260
  - representation schemes 249–251
  - reproduction operators 257–260
- PID tuning problem 304–312
- GA (genetic algorithm) class 268
- GA (genetic algorithms)
  - solving traveling salesman problem 300–304
- galaxy-based search algorithm (GSA) 60
- GALBPs (generalized assembly line balancing problems) 222
- GAT (graph attention network) 416
- Gaussian mutation 278
- gbest PSO (global best PSO) 333
- GbSA (spiral galaxy-based search algorithm) 60
- GCNConv layers 413
- GCNs (graph convolutional networks) 408
- GDL (geometric deep learning) 407, 449
- generate\_combinations function 487
- generational GA 260
- GENERIC\_TABU\_SEARCH 217
- GENERIC\_TABU\_SEARCH method 217
- genes 241
- genetic algorithm (GA) 59
- genetic algorithms 233
  - evolutionary computation
    - biology fundamentals 241
    - theory of evolution 242
  - permutation-based genetic algorithms
    - crossover methods
      - cycle crossover 288
      - edge crossover 286
      - order 1 crossover 287
      - partially mapped crossover 283
    - mutation methods 290
- Python packages for 265
- real-valued genetic algorithms
  - crossover methods 275
    - simple arithmetic crossover method 276
    - simulated binary crossover method 278
    - single arithmetic crossover method 276
    - whole arithmetic crossover method 277
- genetic algorithms
  - evolutionary computation 241–246
    - evolutionary computation 243–246
  - Gray-coded genetic algorithms 272–274
  - implementing in Python 262–269
  - multi-objective optimization 291–298
  - permutation-based genetic algorithms 282–291
    - crossover methods 283–289
  - political districting problem 312–318
  - population-based metaheuristic
    - algorithms 234–241
  - real-valued genetic algorithms 275–282
    - mutation methods 278–282
  - solving traveling salesman problem 300–304
  - traveling salesman problem 300–304
- genetic algorithms (GAs)
  - adaptive GA 298
- genetic operators 244
- genetic programming (GP) 59
- genotype 241
- geometric cooling schedule 170
- geometric deep learning 403
- geometric deep learning (GDL) 407
- gerrymandering 46
- get\_routes() function 218
- gi function 45
- globalBest PSO 360
- global minimum 8
- global optimum 8
- global search 50
- GML (graph machine learning) 407, 409, 449
- GNN (graph neural networks) 408
- Gn point 44
- goal 16
- GP (genetic programming) 59, 234, 245
- gradient ascent 117
- gradient descent 354
- gradient descent algorithm 117
- Gramacy & Lee function 178
- graph embedding 407
- graph machine learning 432–438
  - supervised 432–438

graph machine learning (GML) 407  
 graphs  
   machine learning with 404  
 graphs 63–72  
 graph search 72  
 graph traversal algorithms 74–89  
   breadth-first search 74–83  
   depth-first search 84–89  
 Gray code 272  
 Gray-coded genetic algorithms 272–274  
 GREEDY\_DESCENT method 217  
 grey wolf optimizer (GWO) 59  
 Griewank 1D, 2D, and 3D functions 178  
 Griewank function 178  
 GRUs (gated recurrent units) 421  
 GSA (galaxy-based search algorithm) 60  
 GSA (generalized simulated annealing) 164  
 gSDE (Generalized state dependent  
   exploration) 472  
 GTA (Greater Toronto Area) 28, 190  
 GWO (grey wolf optimizer) 59  
 gym-electric-motor environment 472  
 gym library 474  
 gymnasium 482

## H

Hamming cliff effect 272  
 Hamming cliff problem 273  
 hashmap 214  
 haversine distance formula 376  
 H (embedding or latent space) 407  
 HER (hindsight experience replay) 472  
 heuristic function 53  
 heuristics and metaheuristics 52–58  
 HHs (highway hierarchies) 105  
 highway-env environment 472  
 hill climbing  
   for routing 146  
 hill climbing algorithm 115–121  
 hill climbing (HC) 105  
 Himmelblau function 213  
 HNR (highway-node routing) 105  
 hu embedding 411  
 hv0 embedding 412  
 hv embedding 411  
 hv vector 407  
 hyperbolic tangent activation function 354

hyperedges 67  
 hypergraph 67

## I

IAE (integral absolute error) 307  
 IBH (improved black hole algorithm) 60  
 ICNN'95 (IEEE International conference on  
   neural networks) 494  
 IDDFS (iterative deepening depth-first search) 73  
 ideal points 296  
 IDS (iterative deepening search) 73  
 immediate reaction mechanism 207  
 improved black hole algorithm (IBH) 60  
 Index routing variable 216  
 inductive embedding 408  
 inductive learning 409  
 inertia component 328  
 inertia weight 332, 344, 349, 359  
 informed search algorithms 103, 104  
   for routing problems  
     A\* for routing 149  
     beam search for routing 148  
     hill climbing for routing 146  
   introduction to 104  
 informed search algorithms  
   for routing problems 146–153  
     contraction hierarchies for routing 151–153  
   MST (minimum spanning tree)  
     algorithms 105–114  
   shortest path algorithms 114–146  
     A\* search algorithm 124–146  
     beam search algorithm 121–124  
     hill climbing algorithm 115–121  
 initial temperature 169  
 initial\_temp parameter 178  
 init\_pop function 262  
 insert mutation 290  
 Inspyred (Bio-inspired Algorithms in Python) 266  
 integral term 305  
 intensification 20  
 ints list 262  
 invasive weed optimization (IWO) 60  
 inversion mutation 290  
 IPython.display 482  
 is\_connected() method 111  
 ISE (integral squared error) 307  
 ISPs (ill-structured problems) 16



ITAE (integral time absolute error) 307  
 iterations at each temperature 172  
 ITSE (integral time square error) 307  
 IWO (invasive weed optimization) 60

## J

Jarník-Prim's algorithm 105  
 jMetalPy 266

## K

KH (krill herd) 59  
 k-hop neighborhood 412  
 KL (Kullback-Leibler) divergence 461  
 knapsack problem 202  
 knapsack problem (KP) 426  
 knn strategy 435  
 knn\_strat parameter 436  
 Kohonen map 422  
 krill herd (KH) 59  
 Kruskal's algorithm 105  
 Kullback-Leibler (KL) divergence 461

## L

L2 norm 427  
 Lagrange function 38  
 Lagrange multipliers 38  
 landscape and number of objective functions 32–37  
 Langermann function 178  
 Laplace domain 308  
 large language models (LLMs) 427  
 Latin hypercube sampling 237  
 Latin square 180, 181  
 lazy updates 133  
 lbest PSO (local best PSO) 334  
 LBH (levy flight black hole) 60  
 LB (lower bound) 250  
 LDS (Sobol low-discrepancy sequence) 237  
 learning  
   solving TSP using unsupervised machine learning 438–441  
   supervised  
     summary 449, 495  
   unsupervised  
     summary 449, 495  
   unsupervised machine learning 438–441  
 learning to configure algorithms 425

levy flight black hole (LBH) 60  
 Levy function 178  
 LIFO (last in, first out) 84  
 linear-inverse cooling schedule 170  
 links 404  
 LOA (lion optimization algorithm) 59  
 local search 50, 196, 198  
 locus 241  
 logarithmic cooling schedule 171  
 log function 352  
 log\_interval 443  
 log\_pointer\_scores 447  
 Lorentz distribution 278  
 lower bound 53  
 LPG (policy gradient loss) 457  
 lr (learning rate) 443  
 LS (local search) 196, 197  
 LSTM (long short-term memory) 402, 419  
 LTV (linear time varying) method 345

## M

MAB (multi-armed bandit) 451, 464  
 machine learning  
   machine learning with graphs 404  
 machine learning  
   demystifying 399–403  
     unreasonable effectiveness of data 400–403  
   machine learning for optimization problems 424–427  
   solving TSP using unsupervised machine learning 438–441  
   unsupervised machine learning 438–441  
 machine learning (ML) 351  
 machine learning (ML)  
   supervised learning 427–432  
     function optimization using 427–432  
 magic squares 181  
 make\_dataset method 436  
 management science 9  
 Markov chain (MC) 161  
 MAVs (micro aerial vehicles) 49  
 MaxCut (maximum cut) 471  
 maximal clique (MC) 426  
 maximal independent set (MIS) 426  
 maximum coverage problem (MCP) 426  
 maximum cut (MaxCut) 426  
 max\_iter parameter 178



- max\_iter\_per\_temp parameter 178
  - max-min ant system 375
  - max\_samples 442
  - MC (Markov chain) 161
  - MDP (Markov decision process) 451, 452
    - reinforcement learning 453
  - MEALPY 176
  - MEALPY library 392
  - MEALPY (population-based meta-heuristic algorithms) 267
  - mealpy.swarm\_based.ABC module 392
  - memory structure 201–205
  - merit function 10
  - message passing 410, 411
  - metaheuristic algorithms 158
  - metaheuristics 53
  - Metaheuristics
    - From Design to Implementation (Talbi) 235
  - midpoint attribute 151
  - MineProbe wheel design problem 43
  - minimize function 38, 268, 281
  - minimum spanning tree (MST) algorithm 72
  - minimum vertex cover (MVC) 426
  - Minkowski distance 349
  - min\_samples 442
  - MIS (maximal independent set) 471
  - Mitchell, Thomas 399
  - MLACO (Machine Learning for Combinatorial Optimization) 427
  - MLBH (multiple population levy black hole) 60
  - ML in conjunction with optimization
    - algorithms 426
  - ML (machine learning) 351, 399
  - ML (machine learning)
    - supervised graph machine learning 432–438
  - MLP (multilayer perceptron) 478
  - MLP (multilayer perceptron) policy 482
  - MlpPolicy 478, 482
  - MLPs (multilayer perceptrons) 402
  - Mlrose (Machine Learning, Randomized Optimization and Search) 267
  - MMAS (max-min ant system) 375
  - MMAS (max-min ant system) algorithm 375
  - mobile\_env 482
  - mobile networks
    - autonomous coordination in using PPO 481–485
  - modal model 201
  - model 414, 435, 444, 446
  - model-based RL (MBRL) 458
  - model-free RL (MFRL) 458
  - model object 392
  - model variable 436
  - module attribute 436
  - MOEA/D (multi-objective evolutionary algorithm based on decomposition) 266, 293
  - MOEAs (multi-objective evolutionary algorithms) 267
  - monkey search lion optimization algorithm (LOA) 59
  - mono-objective optimization problem 12
  - motion equations 328
  - MSE (mean squared error) 279, 352
  - MST (minimum spanning tree) 104, 106
  - MST (minimum spanning tree) algorithm 72
  - MST (minimum spanning tree)
    - algorithms 105–114
  - multigraph 64
  - multi-objective optimization 291–298
  - multi-objective optimization algorithms 293
  - multi-objective optimization problem 12
  - multiple population levy black hole (MLBH) 60
  - multiplication 342
  - multi-store model 201
  - mutate method 191
  - mutation 259
  - mutation function 264
  - mutation methods 290
  - mutation methods 278–282
  - mutation operator 244
  - mutation parameter 281
  - mutation\_prob argument 264
  - myopic 456
- ## N
- 
- n\_actions 475
  - nadir points 296
  - natural selection 243
  - nature-inspired algorithms 59, 61
  - navariral.board.chunks\_3 355
  - near-optimal solutions 7
  - negation transformation 249
  - negative log likelihood (NLL) 352
  - neighborhood function 423
  - neighborhoods
    - PSO 333

- neighborhood size 333
- neighborhood structure 196
- neighbors 435
- neighbors parameter 435, 436
- n\_epochs (number of training epochs) 443
- NetLogo 367
- networkx library 380
- neural combinatorial optimization 424
- neural networks
  - with PSO 351–360
- neural networks (NNs) 60
- NeurIPS (Neural Information Processing Systems) conference 427
- New optimization techniques in engineering
  - Studies in fuzziness and soft computing (Clerc) 494
- NLL (negative log likelihood) 352
- NLP (nonlinear programming problem) 42
- NLTV (nonlinear time varying) method 345
- nn.Module 430
- nn.Module class 474, 475
- NNs (neural networks) 60
- nn.TransformerEncoderLayer 443
- NodeIndex 216
- nodes 404
- nondominated solutions 292
- nonlinear functions 351
- np.argmax() function 468
- n\_particles 349
- NPGA (niched-Pareto genetic algorithm) 293
- n-point crossover 258
- n-queen CSP (constraint-satisfaction problem) 34
- n-queen problem 13, 33
- NSGA-III (non-dominated sorting genetic algorithm) 266
- NSGA-II (non-dominated sorting genetic algorithm) 266
- NSGA-II (nondominated sorting genetic algorithm) 293
- NS-PSO (non-dominated sorting particle swarm optimization) 266
- nucleus 241
- number of samples 435
- num\_parents argument 263
- numpy arrays 55
- num\_samples parameter 436
- num\_workers parameter 436

## O

- objective functions 10
  - landscape and number of 32–37
- off-policy RL methods 458
- online delayed pheromone update 369
- onlooker bees 386
- on-policy RL methods 458
- operations research (OR) 9
- operator (successor) 16
- optalgotools package 76
- optimal solutions 7
- optimization
  - ingredients of optimization problems
    - constraints 14
    - decision variables 9
  - PSO (particle swarm optimization)
    - neighborhoods 333
  - search algorithms 20
- optimization
  - heuristics and metaheuristics 52–58
  - ingredients of optimization problems 7–15
    - objective functions 10–13
  - machine learning for optimization
    - problems 424–427
  - well-structured problems vs. ill-structured problems 15–19
  - with reinforcement learning 470–472
- optimization algorithms
  - classification of 52
  - simulated annealing
    - solving Sudoku 180–184
- optimization by prompting (OPRO) 427
- optimization problems
  - classifying 24–50
    - constraints 37–45
    - expected quality and permissible time for solution 45–50
  - landscape and number of objective functions 32–37
  - number and type of decision variables 26–32
- optimization techniques 9
- options dictionary 349
- opts class 434
- order 1 crossover 287
- OriginalABC class 392
- OR (operations research) 9
- OSM (OpenStreetMap) 70

out dictionary 280  
 overlay graph 131

## P

---

PaDE (parameter adaptive differential evolution) 266  
 PAES (Pareto archived evolution strategy) 266, 292  
 Pandana library 153  
 parallel edges 64  
 parameter tuning 205  
 parents argument 264  
 parent selection method 244  
 Pareto frontier 13  
 Pareto optimal solutions 292  
 Pareto optimization 292  
 Pareto optimization approach 13  
 partially mapped crossover 283  
 particle position initialization 332  
 particle swarm optimization 321  
 particle swarm optimization  
   traveling salesman problem  
     solving with PSO 348–351  
 particle swarm optimization (PSO) 59, 324  
 particle velocity initialization 332  
 path\_indices list 381  
 paths array 378  
 PCA (principal component analysis) 402, 424  
 PCA (principle component analysis) 414  
 PDFs (probability density functions) 461  
 peak overshoot 306  
 penguins.owl.owl.chunks\_1 354  
 permutation-based genetic algorithms  
   crossover methods  
     cycle crossover 288  
     edge crossover 286  
     order 1 crossover 287  
     partially mapped crossover 283  
   mutation methods 290  
 permutation-based genetic algorithms 282–291  
   crossover methods 283–289  
 permutation-based PSO 342  
 permutations 26  
 personal best position initialization 332  
 phenotype 241  
 physical annealing 159  
 PIDProblem class 310  
 PID (proportional integral derivative)  
   controller 271, 305, 308  
   PID (proportional integral derivative) controller  
     parameters 275  
   PID (proportional integral derivative) tuning  
     problem 304–312  
   PID tuning problem 307  
 pk (selection probability) 253  
 planning problems 47  
 Platypus 267  
 PLP (plant layout problem) 42  
 P-metaheuristics 159  
 P-metaheuristics (population-based metaheuristic  
   algorithms) 234  
 PMF (probability mass function) 488  
 PointCrossover class 268, 269  
 pointer\_argmaxs 447  
 pointer networks 419  
 points of interest (POI) 69  
 POI (points of interest) 69  
 POIs (points of interest) 190  
 policy gradient 457  
 policy gradient methods 456  
 policy iteration methods 456  
 political districting problem 312–318  
 polynomial mutation 278  
 PolynomialMutation class 268, 269  
 PolynomialMutation operator 281  
 POMDP (partially observable Markov decision  
   process) 454  
 pop\_size argument 262  
 pop\_size parameter 281, 295  
 population argument 263, 264  
 population-based algorithms 158  
 population-based metaheuristic  
   algorithms 234–241  
 population of individuals 244  
 potential energy 44  
 PPO-clip 462  
 PPO-penalty 461  
 PPO (proximal policy optimization) 460, 472  
   PPO-clip 462  
   PPO-penalty 461  
 PPO (proximal policy optimization)  
   autonomous coordination in mobile networks  
     using 481–485  
 PPO (proximal policy optimization)  
   algorithm 479, 484  
   balancing CartPole using 473  
 preprocessing phase 130

principal component analysis (PCA) 424  
 print function 38  
 print\_solution() function 217  
 priority term 133  
 Problem class 268  
 problem\_dict 392  
 problems class 193  
 probs array 358  
 proportional term  $K_p(t)$  305  
 pseudo-random proportional action rule 374  
 PSO (particle swarm optimization) 59, 235, 266, 322, 324, 362, 393  
   adaptive  
     adaptive PSO  
       cognitive component 346  
       inertia weight 344  
       social components 346  
   continuous PSO 326  
     algorithm 326  
   fitness update 330  
   initialization 332  
   motion equations 328  
   neighborhoods 333  
   permutation-based PSO 342  
 PSO (particle swarm optimization)  
   adaptive 343–348  
   binary PSO 340–342  
   neural network training 351–360  
   solving TSP with 348–351  
   swarm intelligence 322–325  
 ptr\_net class 442  
 Ptr-Net (pointer network) 419, 449  
 ptr\_net.py class 443  
 PuLP (Python Linear Programming) library 184  
 p-value 349  
 PyBullet Gym 472  
 pyDOE (Design of Experiments) 237  
 Pyevolve 267  
 PyGAD (Python Genetic Algorithm) 266  
 PyGMO (Python Parallel Global Multi-objective Optimizer) 266  
 PyG (PyTorch Geometric) 413  
 pymoo 239  
 pymoo library 267  
 Pymoo (Multi-objective Optimization in Python) 266  
 pymoo.operators.selection class 257  
 Pyrosm library 153

Python  
   packages for genetic algorithms in 265  
 Python  
   implementing genetic algorithms in 262–269  
 pywrapcp module 217

## Q

---

qa194.tsp 440  
 QAP (quadratic assignment problem) 42  
 QA (quantum annealing) 60, 164  
 QEA (quantum-inspired evolutionary algorithm) 60  
 QGA (quantum-inspired genetic algorithm) 60  
 qk (cumulative probability) 253  
 QoE (quality of experience) 481  
 QP (quadratic programming) 42  
 QPSO (quantum-inspired particle swarm optimization) 60  
 Q-Q plot 354  
 QSE (quantum swarm evolutionary algorithm) 60  
 quadratic complexity 80  
 quantum annealing (QA) 60  
 quantum-inspired evolutionary algorithm (QEA) 60  
 quantum-inspired genetic algorithm (QGA) 60  
 quantum-inspired particle swarm optimization (QPSO) 60  
 quantum swarm evolutionary algorithm (QSE) 60  
 query phase 130  
 quiet 489

## R

---

random.choices function 377  
 random.gauss() function 175  
 random-restart hill climbing 117  
 random selection 257  
 rank-based selection 254  
 Rastrigin function 178  
 ray optimization (RO) 60  
 real-valued genetic algorithms  
   crossover methods 275  
     simple arithmetic crossover method 276  
     simulated binary crossover method 278  
     single arithmetic crossover method 276  
     whole arithmetic crossover method 277  
 real-valued genetic algorithms 275  
   mutation methods 278–282  
 recency-based memory 201

- reciprocal transformation 249
- RegisterTransitCallback() method 217
- regression 402
- reinforcement learning 450
  - balancing CartPole using A2C and PPO 473
  - combinatorial optimization problems 491
  - Markov decision process 452
  - multi-armed bandit 464
  - overview 451
  - summary 496
- reinforcement learning
  - autonomous coordination in mobile networks using PPO 481–485
  - truck selection problem, solving using contextual bandits 485–490
- reinforcement learning (RL) 450
- relaxing node 135
- relu activation function 413
- ReLU (rectified linear unit) 354
- ReLU (rectified linear unit) 352
- Repair class 302
- repeated solution construction heuristic 57
- repeated solution modification heuristic 58
- repeated solution recombination heuristic 58
- repetition argument 189
- representation schemes 249–251
- reproduction operators
  - crossover 257
  - mutation 259
  - new population 259
- reproduction operators 257–260
- reshape() function 335
- resource or time-constrained problems 282
- responsive exploration 197
- results\_sb directory 482
- ride-sharing problem 47
- rise time 306
- RL Baselines3 Zoo 472
- RLOR (reinforcement learning for routing problems) 471
- RL Reach 472
- RL (reinforcement learning) 403, 450
  - actor-critic methods 459
  - MDP (Markov decision process) to 453
  - model-based versus model-free RL 458
  - proximal policy optimization 460
    - PPO-clip 462
    - PPO-penalty 461

- RL (reinforcement learning)
  - optimization with 470–472
- RNN (recurrent neural network) 419, 441
- RNNs (recurrent neural networks) 402
- RO (ray optimization) 60
- rosenbrock\_function 389
- Rosenbrock's function 178
- RoundingRepair class 268, 269
- Route variable 382
- RoutingIndexManager class 217
- RoutingIndexManager object 217
- RoutingModel object 217
- routing problem
  - blind search algorithms and 98–101
- routing problems
  - informed search algorithms for
    - A\* for routing 149
    - beam search for routing 148
    - hill climbing for routing 146
- routing problems
  - informed search algorithms for 146–153
    - contraction hierarchies for routing 151–153
    - solving using tabu search 215–221
- RoutingSearchParameters class 217
- RS (random selection) method 345
- run\_ACO function 378

## S

---

- SAC (soft actor-critic) 472
- SALBP-1 (type 1 simple assembly line balancing problem) 224
- SALBP-2 (type 2 simple assembly line balancing problem) 224
- SALBPs (simple assembly line balancing problems) 222
- sample\_payoff 467
- sample\_payoff() function 468
- sample\_truck\_pmf function 488
- sampling parameter 281
- SA (simulated annealing) 59, 158
  - algorithm
    - annealing process
      - final temperature 172
      - initial temperature 169
      - iterations at each temperature 172
      - temperature decrement 170
    - physical annealing 159

- SA (simulated annealing)
  - algorithm 159–175
    - acceptance probability 165–167
    - adaptation in 172–175
    - annealing process 168–172
    - pseudocode 161–165
  - semi-truck routing problem 190–193
- satisfiability problem (SAT) 426
- SAT (satisfiability problem) 471
- SB3 (Stable Baselines3) 478
- SB3 (Stable-Baselines3) 472
- SBX operator 281
- SBX (simulated binary crossover) 278
- Scatter2DDataset class 442
- Schaffer function 178
- Schwefel function 178
- scipy.optimize 38
- scipy.optimize.anneal 176
- scout bees 386
- scramble mutation 290
- SDS (stochastic diffusion search) 60
- seaborn library 474
- search
  - ingredients of optimization problems
    - constraints 14
    - decision variables 9
- search
  - heuristics and metaheuristics 52–58
  - ingredients of optimization problems 7–15
  - objective functions 10–13
- search algorithms 20
- search algorithms
  - classification of 50–52
- search and optimization 3, 23
  - going from toy problems to real world 6
  - reasons for caring about 5
- search and optimization
  - well-structured problems vs. ill-structured problems 15–19
- search and optimization algorithms
  - nature-inspired algorithms 59
- seed parameter 281
- selection operators
  - elitism 252
  - fitness-proportionate selection 253
  - random selection 257
  - rank-based selection 254
  - stochastic universal sampling 255
  - tournament selection 256
- selection operators 252–257
- select\_parent function 263, 265
- self-adaptive tuning 206
- semi-truck routing problem 190–193
- sensory memory 201
- separable function 178
- Seq2Seq (sequence-to-sequence) 402
- seq2seq (sequence-to-sequence) model 419
- sequential ML
  - pointer networks 419
- SetArcCostEvaluatorOfAllVehicles() method 217
- settling time 306
- SFLA (shuffled frog leaping algorithm) 59
- shortcut edges 145
- shortest path algorithms
  - bidirectional Dijkstra search 96
  - UCS 94
- shortest path algorithms 89, 114
  - A\* search algorithm 124–146
  - beam search algorithm 121–124
  - Dijkstra's search 90–94
  - hill climbing algorithm 115–121
- shortest\_path function 191
- short-term memory 201
- shuffled frog leaping algorithm (SFLA) 59
- side information 470
- sigmoid function 352
- simple arithmetic crossover method 276
- simple hill climbing 117
- simulated annealing 157
  - trajectory-based optimization 158
- simulated annealing
  - function optimization 175–180
  - solving Sudoku 180–184
- SIMULATED\_ANNEALING method 217
- simulated annealing (SA) 59
- simulated annealing (SA)
  - traveling salesman problem and 185–189
- simulated binary crossover method 278
- single arithmetic crossover method 276
- SinglePointCrossover class 268
- singleton array 335
- SI (smoothing index) 224
- SI (swarm intelligence) algorithms 235
- size-1 array 335

- sliding-block problem 75
- slot machine 464
- smallest\_width\_first heuristic 56
- S-metaheuristic algorithms 234
- S-metaheuristics 159
- SMS-EMOA (multi-objective selection based on dominated hypervolume) 293
- social acceleration coefficient 349
- social component 329, 346
- social parameter 359
- social spider optimization (SSO) 59, 324
- SOFM (self-organizing feature map) 422
- softmax activation function 355
- softmax function 352
- solution object 76
- solution/path 16
- solve method 381
- solve() method 392
- SolveWithParameters() method 217
- SOMs (self-organizing maps) 402, 422–424, 449
- space complexity 89
- SPEA2 (strength Pareto evolutionary algorithm) 266, 293
- species {{{PIPE}}} penguin example 354
- sphere\_dist(x, y) function 429
- spiral galaxy-based search algorithm (GbSA) 60
- spring\_layout() method 111
- S-PSO (synchronous PSO) 330
- SSO (social spider optimization) 59, 324
- Stable Baselines3 (SB3) 478
- stagnation 205
- State class 76, 85
- state\_dim 475
- state-value function 456
- static tabu tenure 204
- stats dictionary 477
- steady state 270
- steady-state error 306
- steady-state GA 260
- steepest-ascent hill climbing 117
- stigmergy 364
- stirling function 31
- stochastic algorithms 51
- stochastic diffusion search (SDS) 60
- stochastic hill climbing 117
- stochastic policy 454
- stochastic universal sampling (SUS) 255
- stopping criteria 16
- strong local minimum 8
- strs list 262
- STSP (symmetric TSP) 27
- subtraction 343
- Sudoku 180–184
- SUMO-RL 472
- supervised and unsupervised learning 397
- supervised graph machine learning 432–438
- supervised learning 402
  - AI-empowered daily routines 398
  - attention mechanisms 416
  - finding convex hull 441
  - graph embedding 407
  - machine learning with graphs 404
  - sequential ML
    - pointer networks 419
- supervised learning
  - function optimization using 427–432
  - graph machine learning 432–438
  - machine learning for optimization problems 424–427
  - self-organizing maps 422–424
  - summary 449, 495
- supervised machine learning
  - unsupervised machine learning 438–441
- supervised parameter 436
- survival methods 244
- survival of the fittest 243
- survivor selection 260
- swap mutation 290
- swarm intelligence 322–325
- swarm intelligence algorithms 362
  - ACO
    - simple ACO 370
  - ACO (ant colony optimization)
    - solving TSP with 376
  - ant colony system 374
  - artificial bee colony algorithm 385
  - max-min ant system 375
- swarm intelligence algorithms
  - ant colony optimization 363–366
  - ant colony optimization (ACO) 366–369
  - artificial bee colony algorithm 383–385
- swarm size 333
- SymPy 39
- sympy.stats.DiscreteMarkovChain class 162



**T**


---

- tabu-active moves 203
- tabu search 195
  - algorithm 198
  - solving continuous problems 213
- TABU\_SEARCH 217
- tabu search
  - solving constraint satisfaction problems 207–213
  - solving TSP and routing problems 215–221
- tabu search (TS) 60
- tabu tenure 204
- tactile-gym 472
- TD3 (twin delayed DDPG) 472
- temperature decrement 170
- tensorboard logdir 482
- TicketPrice problem 268
- tile-puzzle problem 75
- time complexity 80, 89
- TNR (transit-node routing) 105
- TOKENS dictionary 442
- to\_numpy() function 359
- torch library 474
- torch.nn.DataParallel 436
- torch.nn.Dropout layer 413
- torch.nn.functional 474
- torch.nn module 474
- torch.optim 474
- to\_undirected function 113
- tournament selection 256
- tour object 382
- tqdm library 474
- tqdm progress bar 431
- training
  - neural networks with PSO 351–360
- train\_loader dataset 444
- trajectory 454
- trajectory-based algorithms 158
- trajectory-based optimization 158
- transductive embedding 408
- transductive learning 409
- TravelingSalesman class 302
- traveling salesman problem (TSP)
  - SA and 185–189
- treasure-hunting mission 52
- TRPO (trust region policy optimization) 461

- truck selection problem, solving using contextual bandits 485–490
- TSC (traffic signal control) 471
- tsp class 215
- tsp\_cost function 349
- tsp\_distance function 349
- tsp object 215
- TSP (traveling salesman problem) 18, 27, 200, 271, 340
- TSP (traveling salesman problem)
  - SA and 185–189
  - solving using tabu search 215–221
  - solving with genetic algorithms 300–304
  - solving with PSO 348–351
  - supervised graph machine learning 432–438
  - unsupervised machine learning 438–441
- TSP (traveling salesperson problem)
  - solving with ACO 376
- TS (tabu search) 60, 195
  - algorithm
    - aspiration criteria 205
  - local search 196
- TS (tabu search)
  - algorithm 197–207
    - adaptation in TS 205–207
    - memory structure 201–205
  - assembly line balancing problem 221–230
- TwoPointCrossover class 268

**U**


---

- UAV\_Navigation\_DRL\_AirSim 472
- UAVs (unmanned aerial vehicles) 49
- UB (upper bound) 250
- ucb(c) function 469
- UCB (upper confidence bound) strategy 466
- UCS (uniform-cost search) 74, 89, 94
- UEs (user equipment) 481
- UGVs (unmanned ground vehicles) 43
- uniform crossover 258
- uniform search 73
- unlabeled data set 354
- unsupervised learning 402
  - AI-empowered daily routines 398
  - attention mechanisms 416
  - finding convex hull 441
  - graph embedding 407
  - machine learning with graphs 404

- unsupervised learning
  - machine learning for optimization problems 424–427
  - self-organizing maps 422–424
  - summary 449, 495
- unsupervised machine learning 438–441
- update\_pheromones function 378
- update\_velocity function 337
- upward graph 135
- use\_cuda 435
- utility function 10

## V

---

- val\_dataset 444
- valley function 387
- val\_loader dataset 444
- variables
  - decision variables 9
- vehicular routing problem (VRP) 426
- velocity clamping 360
- verbose parameter 281
- vertices 404
- visualize function 76
- VLSI (very large-scale integration) 23
- Von Neumann model 334
- Vowpal Wabbit (VW) 470

- VRP (vehicle routing problem) 471
- vtype parameter 280
- VW (Vowpal Wabbit) 470, 486

## W

---

- water cycle algorithm (WCA) 60
- water flow-like algorithm (WFA) 60
- WCA (water cycle algorithm) 60
- weak local minimum 8
- weighted graph 68
- WFA (water flow-like algorithm) 60
- wheel topology 334
- whole arithmetic crossover method 277
- WIP (work in progress) 222
- witness path 131
- wolf search algorithm (WSA) 59
- working memory 201
- WSA (wolf search algorithm) 59, 235
- WSPs (well-structured problems) 15
- WSP (well-structured problem) 75

## X

---

- XOR (exclusive OR) gates 272

## Z

---

- zfill() method 262