

目 次

第 1 章 はじめに	1
1.1 メタヒューリスティクスとは	1
1.2 最適化問題とは	3
1.3 メタヒューリスティクスの基本戦略	6
1.3.1 近傍の利用	6
1.3.2 構築法の利用	8
1.3.3 部品の利用	11
1.3.4 複数の解の保持	12
1.3.5 ランダム性の利用	12
1.3.6 問題の変形	13
1.3.7 探索の履歴の利用	15
第 2 章 代表的なメタヒューリスティクス	17
2.1 局所探索法	17
2.2 多出発局所探索法	21
2.3 反復局所探索法	23
2.4 模擬焼なまし法	25
2.5 禁断探索法	35
2.6 誘導局所探索法	41
2.7 大近傍探索法	44
2.8 探索空間平滑化法と交互平滑化法	46
2.9 部品最適化法	53
2.10 多レベル法	54

2.11	貪欲ランダム適応型探索法	57
2.12	蟻群生法	60
2.13	遺伝的アルゴリズム	61
2.14	散布探索法	64
第 3 章	数理計画とメタヒューリスティクスの融合	69
3.1	分枝限定法	69
3.2	なぜ融合が必要か?	75
3.3	変数固定法	77
3.4	打ち切り分枝限定法と飛び込み法	79
3.5	緩和固定法	79
3.6	容量スケールリング法	82
3.7	MIP 近傍局所探索法	84
3.8	局所分枝法	85
3.9	MIP 併合法	87
第 4 章	応 用	89
4.1	グラフ分割問題	90
4.1.1	問題の定義と定式化	91
4.1.2	近 傍	93
4.1.3	補助メモリによる高速化	94
4.1.4	ペナルティ関数法	96
4.1.5	擬似実行可能解探索法	98
4.1.6	模擬焼なまし法	99
4.1.7	禁断探索法	100
4.1.8	Python による実装	101
4.2	最大安定集合問題	114
4.2.1	問題の定義と定式化	114
4.2.2	近 傍	116
4.2.3	禁断探索法	118
4.2.4	平坦探索	119

4.2.5	Python による実装	120
4.3	グラフ彩色問題	130
4.3.1	問題の定義と定式化	131
4.3.2	近 傍	133
4.3.3	補助メモリによる高速化	137
4.3.4	禁断探索法	138
4.3.5	遺伝的アルゴリズムと禁断探索法の融合	138
4.3.6	Python による実装	141
4.4	巡回セールスマン問題	152
4.4.1	問題の定義と定式化	152
4.4.2	近 傍	155
4.4.3	局所探索法	156
4.4.4	初回改善と最良改善	159
4.4.5	Python による実装	161
4.5	2 次割当問題	164
4.5.1	問題の定義と定式化	165
4.5.2	近 傍	170
4.5.3	目的関数の差の計算	170
4.5.4	禁断探索法	172
4.5.5	Python による実装	172
4.6	多制約ナップサック問題	176
4.6.1	問題の定義と定式化	177
4.6.2	禁断探索法	178
4.6.3	線形計画緩和を用いた解法	181
4.6.4	Python による実装	182
4.7	数分割問題	188
4.7.1	問題の定義と定式化	189
4.7.2	差分法	190
4.7.3	分割数が 3 以上の場合	191
4.7.4	Python による実装	194

付録 Python 概説	201
A.1 なぜ Python か？	201
A.2 データ型	203
A.2.1 数	203
A.2.2 文字列	203
A.2.3 リスト	204
A.2.4 タプル	206
A.2.5 辞書	207
A.2.6 集合	207
A.3 演算子	208
A.4 制御フロー	209
A.4.1 分岐	209
A.4.2 反復	210
A.5 関数	213
A.6 モジュール	214
A.6.1 擬似乱数発生モジュール random	215
A.6.2 ヒープモジュール heapq	215
参考文献	217
索引	223