進捗報告

1 今週行ったこと

- 評価用ソースコードを Python に書き換えたものと C++のコードの比較
- CMA-ES アルゴリズム（Deap/cma）を用いて実行可能解の探索

1.1 評価用コードの比較

Python コードでは，表 1 にある有効桁数の範囲では全く同じ値になったが，コマンドプロンプトでは C++コードでは制約違反合計値はオーダーはあっているものの誤差が見られた．Cygwin では一致したため，コードには問題がないと思われる．

表 1: 評価用コードの値

実行環境	目的関数値	制約違反合計値
solution_x_1	399635.845	6.43e-12
C++(cmd)	3999635.845	6.26e-12
C++(Cygwin)	399635.845	6.43e-12
Python	399635.845	6.43e-12

1.2 CMA-ES アルゴリズム（Deap/cma）を用いて実行可能解の探索

Deap の cma を用いて実装した．

元の問題の目的関数を F，違反関数合計値を V，ペナルティ関数の重み ρ とすると，deap 上での目的関数 F’ は次の式で表される．

$$F' = F + \rho\ V$$
$$\rho\ = 10^{12}$$

実行可能解は出たが，制約違反を重視するあまり，目的関数値は 490 万程度と，既知の解より大幅に増えていた（既知の解の目的関数値は 400 万程度）．

また，x は全て正であるという制限を課していないため，制約違反を潰すように探索するのに大幅な時間がかかった上，負の値を含む結果となった．既知の解と今回得られた解の変数 x を比較すると，ガスタービンのガス消費量が変数 x に含まれるが，ガスタービンを全く動かさないという結果になってしまった．

ガスタービン一台，ボイラ一台，ターボ冷凍機一台，蒸気吸収式冷凍機二台のプラントの 24 時刻運用計画であるが，ガスタービンを動かさずに購入電力量とボイラからの蒸気生成量で賄ってしまっている．

2 次回行うこと

- x の取りうる値を定める．（探索の際に既知解の利用はしていいのか．）
- どうやって，ガスタービンを動かさないモデルにならないようにするか考える．