Finding Attractive Technical Patterns

- Attractive technical patterns in stock market
  - Profitable
  - Human interpretable
  - Frequent
- Use genetic programming (GP) to evolve attractive technical patterns

**Figure 1:** Genetic programming for finding attractive technical patterns

\[
E_k(r) = \frac{1}{|R(r)|} \sum_{(i,j) \in R(r)} P_r(i, j + k) \quad P_r(i, j)
\]

**Figure 2:** Expected earning rate of pattern \( r \) after \( k \) trading days

\[
f(r) = \begin{cases} 
\frac{1}{n} \sum_{k=1}^{n} E_k(r) & \text{if } |r| < M, |R(r)| \geq m \\
0 & \text{otherwise}
\end{cases}
\]

**Figure 3:** Fitness function for modular GP

Log-Optimal Portfolio Selection

- Maximizing the expected log investment return of a portfolio
  \[
b = (b_1, b_2, \ldots, b_m)^T, \quad b_\geq 0, \quad \sum b_i = 1
\]
  \[
X = (X_1, X_2, \ldots, X_m)^T \sim F(x), \quad x \in \mathbb{R}^m
\]

**Figure 4:** A portfolio and stock vector

\[
W(b) = E \ln b^T X = \int \ln b^T x dF(x)
\]

**Figure 5:** Expected log investment return of a portfolio

- We wish to find an optimal portfolio \( b \)
  - Various approaches are available
- Iterative algorithms approach
  - Need to calculate \( W(b) \) but requires complicated integration
  - Can be overcome by the method of sampling

\[
X_1, \ldots, X_m \sim p \quad \text{i.i.d.}
\]

\[
\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} f(X_i)
\]

**Figure 6:** A basic Monte Carlo estimate of \( Ef(X) \)

Parallelization of Computing Expected Return

- Do the following for each thread;
  - Sample \( X \) from \( F(x) \);
  - Calculate \( \ln b^T X \);
  - Perform parallel reduction;
- Until some accuracy criteria;

**Algorithm 1:** Parallel computation of \( W(b) \)

- Computing \( W(b) \) fast enough allows us to use iterative methods for the optimization

Challenges

- Finding parallelizable component of a program
- Speed and accuracy trade off
  - Single precision vs. double precision
  - More complicated than single core environment
  - May even lead to better performance and better accuracy all at the same time
- Parallelization itself is still tedious and difficult
  - Carefully planning memory access pattern
  - Exploiting the parallel memory architecture
  - Concise representation of data
  - Host/device memory data transfer pattern
  - Exploiting parallel program patterns
  - Fine tuning using device level knowledge

\(^a\) Compared to single core version of the program
\(^b\) NVIDIA GTX 690