GPU acceleration in early-exercise option valuation

Álvaro Leitao and Cornelis W. Oosterlee

Financial Mathematics and Supercomputing

A Coruña - September 26, 2018

Motivation

- Efficient valuation of early-exercise options.
- Novel method: combination of successful previous ideas.
- Originally introduced by Jain and Oosterlee in 2013.
- Multi-dimensional early-exercise option contracts.
- Increase the dimensionality.
- The technique becomes very expensive.
- Solution: parallelization of the method.
- GPU computing (GPGPU).

Outline

Definitions

Option

A contract that offers the buyer the right, but not the obligation, to buy (call) or sell (put) a financial asset at an agreed-upon price (the strike price) during a certain period of time or on a specific date (exercise date). Investopedia.

Option price

The fair value to enter in the option contract. In other words, the (discounted) expected value of the contract.

$$V_t = D_t \mathbb{E}\left[f(S_t)\right]$$

where f is the *payoff* function, S the underlying asset, t the exercise time and D_t the discount factor.

Definitions (II)

Pricing techniques

- Stochastic process, S_t, governing by a SDE.
- Simulation: Monte Carlo method.
- PDEs: Feynman-Kac theorem.
- Fourier inversion techniques: Characteristic function.

Types of options - Exercise time

- European: End of the contract, $t = T$.
- American: Anytime, $t \in [0, T]$.
- Bermudan: Some predefined times, $t \in \{t1, \ldots, t_M\}$
- Many others: Asian, barrier, ...

Definitions (III)

Early-exercise option price

- American:

$$V_t = \sup_{t \in [0,T]} D_t \mathbb{E}\left[f(S_t)\right].$$

- Bermudan:

$$V_t = \sup_{t \in \{t1,\ldots,t_M\}} D_t \mathbb{E}\left[f(S_t)\right].$$

Pricing early-exercise options

- PDEs: Hamilton-Jacobi-Bellman equation.
- Fourier inversion techniques: low dimensions.
- Simulation:
  - Least-squares method (LSM), Longstaff and Schwartz.
  - Stochastic Grid Bundling method (SGBM) [JO15].

Basket Bermudan Options

- Right to exercise at a set of times: $t \in \{t_0 = 0, \dots, t_m, \dots, t_M = T\}$.
- d-dimensional underlying process: $\mathbf{S}_t = (S_t^1, \dots, S_t^d) \in \mathbb{R}^d$.
- Driven by a system of SDE in the form:

$$
\begin{aligned}
\mathrm{d}S_t^1 &= \mu_1(\mathbf{S}_t)\mathrm{d}t + \sigma_1(\mathbf{S}_t)\mathrm{d}W_t^1, \\
\mathrm{d}S_t^2 &= \mu_2(\mathbf{S}_t)\mathrm{d}t + \sigma_2(\mathbf{S}_t)\mathrm{d}W_t^2, \\
&\vdots \\
\mathrm{d}S_t^d &= \mu_d(\mathbf{S}_t)\mathrm{d}t + \sigma_d(\mathbf{S}_t)\mathrm{d}W_t^d,
\end{aligned}
$$

where $W_t^\delta, \delta = 1, 2, \dots, d$, are correlated standard Brownian motions. The instantaneous correlation coefficient between W_t^i and W_t^j is $\rho_{i,j}$.

Basket Bermudan Options (II)

- Intrinsic value of the option: $h_t := h(\mathbf{S}_t)$.
- The value of the option at the terminal time T:

$$V_T(\mathbf{S}_T) = f(\mathbf{S}_T) = \max(h(\mathbf{S}_T), 0).$$

- The conditional continuation value Q_{t_m}, i.e. the discounted expected payoff at time t_m:

$$Q_{t_m}(\mathbf{S}_{t_m}) = D_{t_m}\mathbb{E}\left[V_{t_{m+1}}(\mathbf{S}_{t_{m+1}})|\mathbf{S}_{t_m}\right].$$

- The Bermudan option value at time t_m and state $\mathbf{S}_{t_m}$:

$$V_{t_m}(\mathbf{S}_{t_m}) = f(\mathbf{S}_T) = \max(h(\mathbf{S}_{t_m}), Q_{t_m}(\mathbf{S}_{t_m})).$$

- Value of the option at the initial state $\mathbf{S}_{t_0}$, i.e. $V_{t_0}(\mathbf{S}_{t_0})$.

Basket Bermudan options scheme

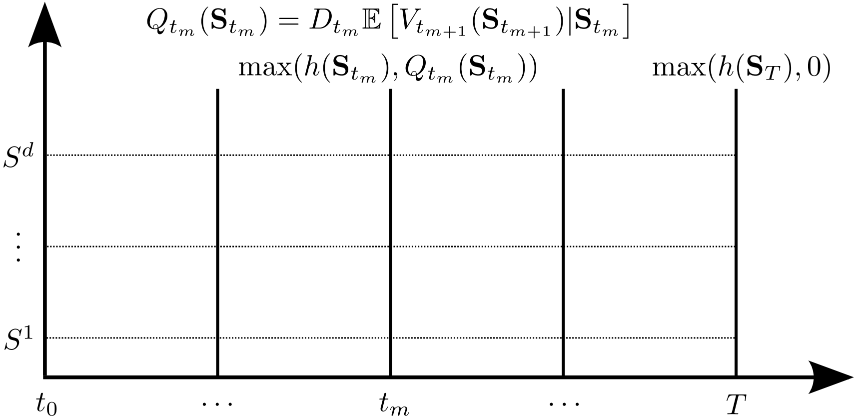


Figure: d-dimensional Bermudan option

Stochastic Grid Bundling Method

- Dynamic programming approach.
- Simulation and regression-based method.
- Forward in time: Monte Carlo simulation.
- Backward in time: Early-exercise policy computation.
- Step I: Generation of stochastic grid points

$$\{\mathbf{S}_{t_0}(n), \ldots, \mathbf{S}_{t_M}(n)\}, \; n = 1, \ldots, N.$$

- Step II: Option value at terminal time $t_M = T$

$$V_{t_M}(\mathbf{S}_{t_M}) = \max(h(\mathbf{S}_{t_M}), 0).$$

Stochastic Grid Bundling Method (II)

- Backward in time, t_m, $m \leq M$,:
- Step **III**: Bundling into ν non-overlapping sets or partitions

$$\mathcal{B}_{t_{m-1}}(1), \ldots, \mathcal{B}_{t_{m-1}}(\nu)$$

- Step IV: Parameterizing the option values

$$Z(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m}) \approx V_{t_m}(\mathbf{S}_{t_m}).$$

- Step V: Computing the continuation and option values at t_{m-1}

$$\widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \mathbb{E}[Z(\mathbf{S}_{t_m}, \alpha^{\beta}_{t_m})|\mathbf{S}_{t_{m-1}}(n)].$$

  The option value is then given by:

$$\widehat{V}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \max(h(\mathbf{S}_{t_{m-1}}(n)), \widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n))).$$

Bundling

- Original: Iterative process (K-means clustering).
- Problems: Too expensive (time and memory) and distribution.
- New technique: Equal-partitioning. Efficient for parallelization.
- Two stages: sorting and splitting.

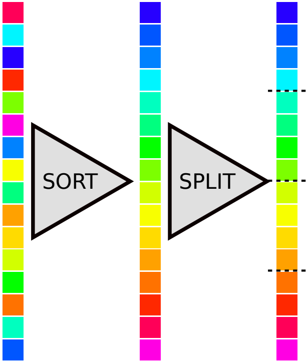


Figure: Equal partitioning scheme

Parametrizing the option value

- Basis functions $\phi_1, \phi_2, \ldots, \phi_K$.
- In our case, $Z\left(\mathbf{S}_{t_m}, \alpha_{t_m}^{\beta}\right)$ depends on $\mathbf{S}_{t_m}$ only through $\phi_k(\mathbf{S}_{t_m})$:

$$Z\left(\mathbf{S}_{t_m}, \alpha_{t_m}^{\beta}\right) = \sum_{k=1}^{K} \alpha_{t_m}^{\beta}(k)\phi_k(\mathbf{S}_{t_m}).$$

- Computation of $\alpha_{t_m}^{\beta}$ (or $\widehat{\alpha}_{t_m}^{\beta}$) by least squares regression.
- The $\alpha_{t_m}^{\beta}$ determines the early-exercise policy.
- The continuation value:

$$\begin{aligned}\widehat{Q}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) &= D_{t_{m-1}}\mathbb{E}\left[\left(\sum_{k=1}^{K}\widehat{\alpha}_{t_m}^{\beta}(k)\phi_k(\mathbf{S}_{t_m})\right)|\mathbf{S}_{t_{m-1}}\right] \\ &= D_{t_{m-1}}\sum_{k=1}^{K}\widehat{\alpha}_{t_m}^{\beta}(k)\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}}\right].\end{aligned}$$

Basis functions

- Choosing ϕ_k: the expectations $\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m})|\mathbf{S}_{t_{m-1}}\right]$ should be easy to calculate.
- The intrinsic value of the option, $h(\cdot)$, is usually an important and useful basis function. For example:
  - Geometric basket Bermudan:

$$h(\mathbf{S}_t) = \left(\prod_{\delta=1}^{d} S_t^{\delta}\right)^{\frac{1}{d}}$$

  - Arithmetic basket Bermudan:

$$h(\mathbf{S}_t) = \frac{1}{d}\sum_{\delta=1}^{d} S_{t_m}^{\delta}$$

- For $\mathbf{S}_t$ following a GBM: expectations analytically available.

Estimating the option value

- SGBM has been developed as *duality-based method*.
- Provide two estimators (confidence interval).
- *Direct estimator* (high-biased estimation):

$$\widehat{V}_{t_{m-1}}(\mathbf{S}_{t_{m-1}}(n)) = \max\left(h\left(\mathbf{S}_{t_{m-1}}(n)\right), \widehat{Q}_{t_{m-1}}\left(\mathbf{S}_{t_{m-1}}(n)\right)\right),$$

$$\mathbb{E}[\widehat{V}_{t_0}(\mathbf{S}_{t_0})] = \frac{1}{N}\sum_{n=1}^{N}\widehat{V}_{t_0}(\mathbf{S}_{t_0}(n)).$$

- *Path estimator* (low-biased estimation):

$$\widehat{\tau}^*\left(\mathbf{S}(n)\right) = \min\{t_m \,:\, h\left(\mathbf{S}_{t_m}(n)\right) \geq \widehat{Q}_{t_m}\left(\mathbf{S}_{t_m}(n)\right),\ m = 1, \ldots, M\},$$

$$v(n) = h\left(\mathbf{S}_{\widehat{\tau}^*(\mathbf{S}(n))}\right),$$

$$\underline{V}_{t_0}(\mathbf{S}_{t_0}) = \lim_{N_{\mathrm{L}}}\frac{1}{N_{\mathrm{L}}}\sum_{n=1}^{N_{\mathrm{L}}} v(n).$$

SGBM - schematic algorithm

Data: $\mathbf{S}_{t_0}, X, \mu_\delta, \sigma_\delta, \rho_{i,j}, T, N, M$
Pre-Bundling (only in k-means case).
Generation of the grid points (Monte Carlo). Step I.
Option value at terminal time $t = M$. Step II.
for *Time* $t = (M-1)\dots 1$ **do**
    Bundling. Step III.
    for *Bundle* $\beta = 1\dots\nu$ **do**
        Exercise policy (Regression). Step IV.
        Continuation value. Step V.
        Direct estimator. Step V.

Generation of the grid points (Monte Carlo). Step I.
Option value at terminal time $t = M$. Step II.
for *Time* $t = (M-1)\dots 1$ **do**
    Bundling. Step III.
    for *Bundle* $\beta = 1\dots\nu$ **do**
        Continuation value. Step V.
        Path estimator. Step V.

Continuation value computation: new approach

- More generally applicable. More involved models or options.
- First discretize, then derive the *discrete* characteristic function.

$$
\begin{aligned}
S^1_{t_{m+1}} &= S^1_{t_m} + \mu_1(\mathbf{S}_{t_m})\Delta t + \sigma_1(\mathbf{S}_{t_m})\Delta\tilde{W}^1_{t_{m+1}},\\
S^2_{t_{m+1}} &= S^2_{t_m} + \mu_2(\mathbf{S}_{t_m})\Delta t + \rho_{1,2}\sigma_2(\mathbf{S}_{t_m})\Delta\tilde{W}^1_{t_{m+1}} + L_{2,2}\sigma_2(\mathbf{S}_{t_m})\Delta\tilde{W}^2_{t_{m+1}},\\
&\dots\\
S^d_{t_{m+1}} &= S^d_{t_m} + \mu_d(\mathbf{S}_{t_m})\Delta t + \rho_{1,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}^1_{t_{m+1}} + L_{2,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}^2_{t_{m+1}} + \cdots + L_{d,d}\sigma_d(\mathbf{S}_{t_m})\Delta\tilde{W}^d_{t_{m+1}},
\end{aligned}
$$

- By definition, the d-variate discrete characteristic function:

$$
\begin{aligned}
\psi_{\mathbf{S}_{t_{m+1}}}(u_1, u_2, \dots, u_d|\mathbf{S}_{t_m}) &= \mathbb{E}\left[\exp\left(\sum_{j=1}^{d} iu_j S^j_{t_{m+1}}\right)|\mathbf{S}_{t_m}\right]\\
&= \mathbb{E}\left[\exp\left(\sum_{j=1}^{d} iu_j\left(S^j_{t_m} + \mu_j(\mathbf{S}_{t_m})\Delta t + \sigma_j(\mathbf{S}_{t_m})\sum_{k=1}^{j} L_{k,j}\Delta\tilde{W}^k_{t_{m+1}}\right)\right)|\mathbf{S}_{t_m}\right]\\
&= \exp\left(\sum_{j=1}^{d} iu_j\left(S^j_{t_m} + \mu_j(\mathbf{S}_{t_m})\Delta t\right)\right)\cdot\prod_{k=1}^{d}\left(\mathbb{E}\left[\exp\left(\sum_{j=k}^{d} iu_j L_{k,j}\sigma_j(\mathbf{S}_{t_m})\Delta\tilde{W}^k_{t_{m+1}}\right)\right]\right)\\
&= \exp\left(\sum_{j=1}^{d} iu_j\left(S^j_{t_m} + \mu_j(\mathbf{S}_{t_m})\Delta t\right)\right)\cdot\prod_{k=1}^{d}\left(\psi_{\mathcal{N}(0,\Delta t)}\left(\sum_{j=k}^{d} u_j L_{k,j}\sigma_j(\mathbf{S}_{t_m})\right)\right),
\end{aligned}
$$

Continuation value computation: new approach

- Joint moments of the product:

$$
\begin{aligned}
M_{\mathbf{S}_{t_{m+1}}} &= \mathbb{E}\left[\left(S^1_{t_{m+1}}\right)^{c_1}\left(S^2_{t_{m+1}}\right)^{c_2}\cdots\left(S^d_{t_{m+1}}\right)^{c_d} |\mathbf{S}_{t_m}\right] \\
&= (-i)^{c_1+c_2+\cdots+c_d}\left[\frac{\partial^{c_1+c_2+\cdots+c_d}\psi_{\mathbf{S}_{t_{m+1}}}(\mathbf{u}|\mathbf{S}_{t_m})}{\partial u_1^{c_1}\partial u_2^{c_2}\cdots\partial u_d^{c_d}}\right]_{\mathbf{u}=0}.
\end{aligned}
$$

- So, if the basis functions are the product of asset processes:

$$
\phi_k(\mathbf{S}_{t_m}) = \left(\prod_{\delta=1}^{d} S^{\delta}_{t_m}\right)^{k-1}, \; k=1,\dots,K,
$$

- This approximation is, in general, worse than the analytic one.
- Feasible thank to the GPU implementation: time steps $\uparrow\uparrow$.

Parallel SGBM on GPU

- NVIDIA CUDA platform.
- Parallel strategy: two parallelization stages:
  - Forward: Monte Carlo simulation.
  - Backward: Bundles $\rightarrow$ Oportunity of parallelization.
- Novelty in early-exercise option pricing methods.
- Other methods: dependency and load-balancing problems.
- More bundles $\rightarrow$ more paths.
- For high dimensions: huge amount of data $(N \times M \times d)$.
- Efficient use of memory is required.

Parallel SGBM on GPU - Forward in time

- One GPU thread per Monte Carlo simulation.
- Random numbers "on the fly": cuRAND library.
- Compute intermediate results:
  - Expectations.
  - Intrinsic value of the option.
  - Equal-partitioning: sorting criterion calculations.
- Intermediate results in the registers: fast memory access.
- Original bundling: all the data still necessary.

Parallel SGBM on GPU - Forward in time

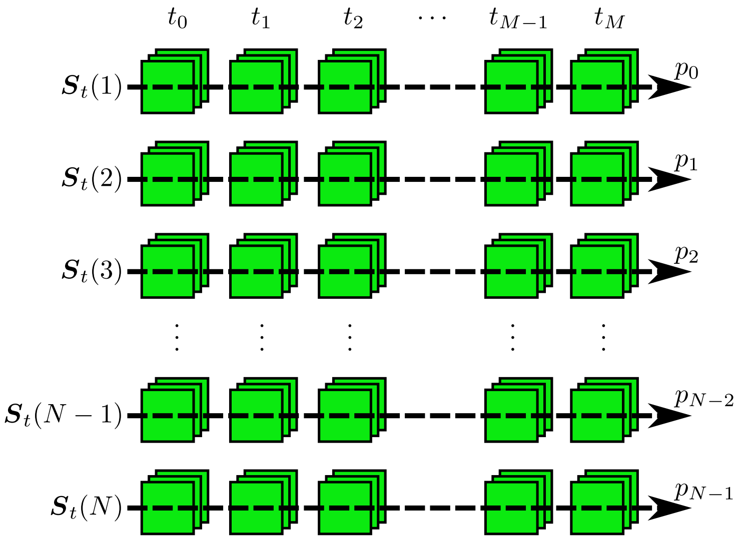


Figure: SGBM Monte Carlo

Parallel SGBM on GPU - Backward in time

- One parallelization stage per exercise time step.
- Sort w.r.t bundles: efficient memory access.
- Parallelization in bundles.
- Each bundle calculations (option value and early-exercise policy) in parallel.
- All GPU threads collaborate in order to compute the continuation value.
- Path estimator: One GPU thread per path (the early-exercise policy is already computed).
- Final reduction: Thrust library.

Parallel SGBM on GPU - Bundling

- Two implementations → K-means vs. Equal-partitioning.
- K-means clustering:
  - K-means: sequential parts.
  - K-means: transfers between CPU and GPU cannot be avoided.
  - K-means: all data need to be stored.
  - K-means: Load-balancing.
- Equal-partitioning:
  - Equal-partitioning: fully parallelizable.
  - Sorting library, CUDPP (Radix sort): kernel-level API.
  - Equal-partitioning: No transfers.
  - Equal-partitioning: efficient memory use.

Parallel SGBM on GPU - Backward in time

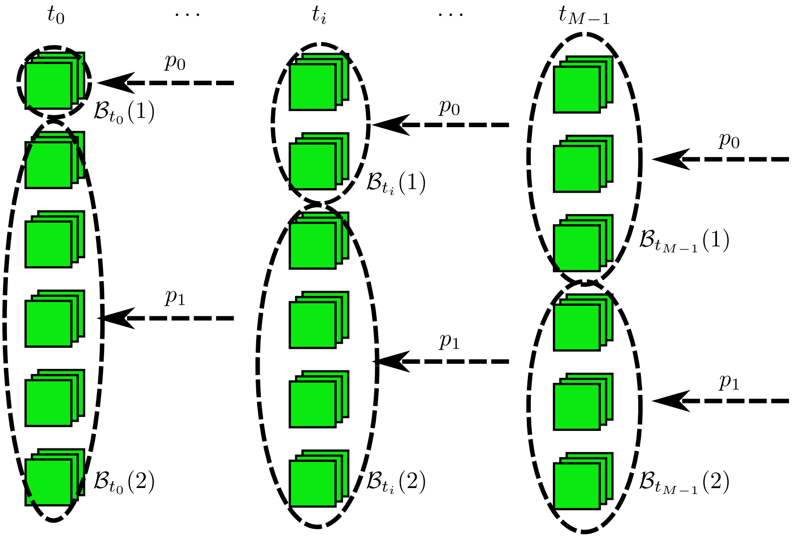


Figure: SGBM backward stage

Parallel SGBM on GPU - Backward in time

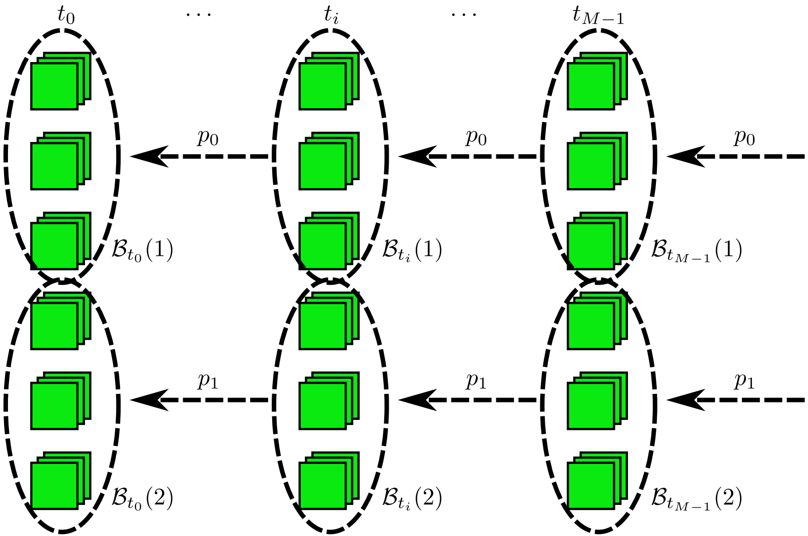


Figure: SGBM backward stage

Parallel SGBM on GPU - Schematic algorithm

Algorithm 1: Parallel SGBM.

```
Data: S_t0, X, μ_δ, σ_δ, ρ_i,j, T, N, M
// Generation of the grid points (Monte Carlo). Step I.
// Option value at terminal time t = M . Step II.
[payoffData, critData, expData] = MonteCarloGPU(S_t0, X, μ_δ, σ_δ, ρ_i,j, T, N, M);
for Time t = M . . . 1 do
    // Bundling. Step III.
    SortingGPU(critData[t-1]);
    begin CUDAThread per bundle β = 1 . . . ν
        α_t^β = LeastSquaresRegression(payoffData[t]); // Exercise policy (Regression). Step IV.
        CV = ContinuationValue(α_t^β, expData[t-1]); // Continuation value. Step V.
        DE = DirectEstimator(CV, payoffData[t-1]); // Direct estimator. Step V.
return DE;
// Generation of the grid points (Monte Carlo). Step I.
// Option value at terminal time t = M . Step II.
[payoffData, critData, expData] = MonteCarloGPU(S_t0, X, μ_δ, σ_δ, ρ_i,j, T, N, M);
for Time t = M . . . 1 do
    SortingGPU(critData[t-1]); // Bundling. Step III.
    begin CUDAThread per path n = 1 . . . N
        CV[n] = ContinuationValue(α_t^β, expData[t-1]); // Continuation value. Step V.
        PE[n] = PathEstimator(CV[n], payoffData[t-1]); // Path estimator. Step V.
return PE;
```

Results

- Accelerator Island system of Cartesius Supercomputer.
  - Intel Xeon E5-2450 v2.
  - NVIDIA Tesla K40m.
  - C-compiler: GCC 4.4.7.
  - CUDA version: 5.5.
- Geometric and arithmetic basket Bermudan put options: $\mathbf{S}_{t_0} = (40, \ldots, 40) \in \mathbb{R}^d$, $X = 40$, $r_t = 0.06$, $\sigma = (0.2, \ldots, 0.2) \in \mathbb{R}^d$, $\rho_{ij} = 0.25$, $T = 1$ and $M = 10$.
- Basis functions: $K = 3$.
- Multi-dimensional Geometric Brownian Motion:

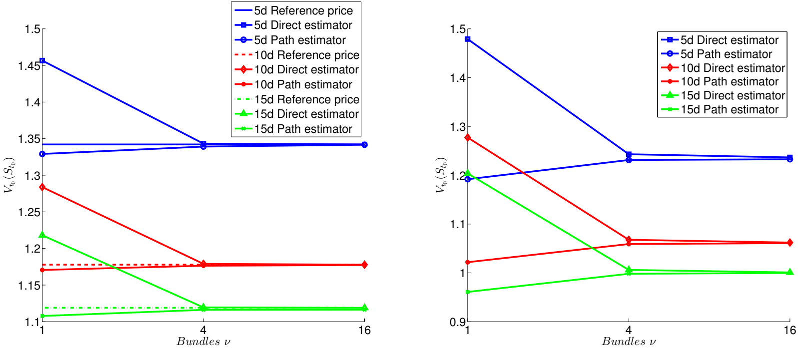
$$\mu_\delta(\mathbf{S}_t) = r_t S_t^\delta, \quad \sigma_\delta(\mathbf{S}_t) = \sigma_\delta S_t^\delta, \quad \delta = 1, 2, \ldots, d,$$

- New approach: Euler discretization, $\delta t = T/M$, CEV model:

$$\mu_\delta(\mathbf{S}_t) = r_t S_t^\delta, \quad \sigma_\delta(\mathbf{S}_t) = \sigma_\delta \left(S_t^\delta\right)^\gamma, \quad \delta = 1, 2, \ldots, d,$$

with $\gamma \in [0, 1]$.

Equal-partitioning: convergence test



(a) Geometric basket put option (b) Arithmetic basket put option

Figure: Convergence with equal-partitioning bundling technique. Test configuration: $N = 2^{18}$ and $\Delta t = T/M$.

Speedup - stages

	Geometric basket Bermudan option					
	k-means			equal-partitioning		
	MC	DE	PE	MC	DE	PE
C	82.42	234.37	203.77	101.77	41.48	59.16
CUDA	1.04	18.69	12.14	0.63	4.66	1.29
Speedup	79.25	12.88	16.78	161.54	8.90	45.86
	Arithmetic basket Bermudan option					
	k-means			equal-partitioning		
	MC	DE	PE	MC	DE	PE
C	78.86	226.23	203.49	79.22	39.64	58.65
CUDA	1.36	17.89	11.74	0.83	4.14	1.20
Speedup	57.98	12.64	17.33	95.44	9.57	48.87

Table: SGBM stages time (s) for the C and CUDA versions. Test configuration: $N = 2^{22}$, $\Delta t = T/M$, $d = 5$ and $\nu = 2^{10}$.

Speedup - total

	Geometric basket Bermudan option					
	k-means			equal-partitioning		
	$d = 5$	$d = 10$	$d = 15$	$d = 5$	$d = 10$	$d = 15$
C	604.13	1155.63	1718.36	303.26	501.99	716.57
CUDA	35.26	112.70	259.03	8.29	9.28	10.14
Speedup	17.13	10.25	6.63	36.58	54.09	70.67
	Arithmetic basket Bermudan option					
	k-means			equal-partitioning		
	$d = 5$	$d = 10$	$d = 15$	$d = 5$	$d = 10$	$d = 15$
C	591.91	1332.68	2236.93	256.05	600.09	1143.06
CUDA	34.62	126.69	263.62	8.02	11.23	15.73
Speedup	17.10	10.52	8.48	31.93	53.44	72.67

Table: SGBM total time (s) for the C and CUDA versions. Test configuration: $N = 2^{22}$, $\Delta t = T/M$ and $\nu = 2^{10}$.

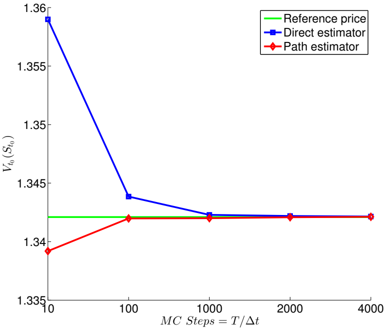
Speedup - High dimensions

	Geometric basket Bermudan option					
	$\nu = 2^{10}$			$\nu = 2^{14}$		
	$d = 30$	$d = 40$	$d = 50$	$d = 30$	$d = 40$	$d = 50$
C	337.61	476.16	620.11	337.06	475.12	618.98
CUDA	4.65	6.18	8.08	4.71	6.26	8.16
Speedup	72.60	77.05	76.75	71.56	75.90	75.85

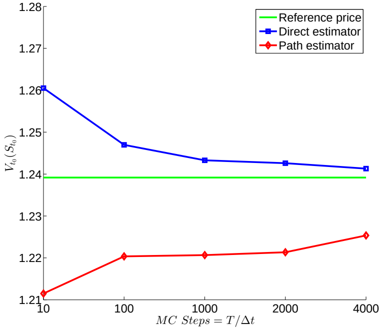
	Arithmetic basket Bermudan option					
	$\nu = 2^{10}$			$\nu = 2^{14}$		
	$d = 30$	$d = 40$	$d = 50$	$d = 30$	$d = 40$	$d = 50$
C	993.96	1723.79	2631.95	992.29	1724.60	2631.43
CUDA	11.14	17.88	26.99	11.20	17.94	27.07
Speedup	89.22	96.41	97.51	88.60	96.13	97.21

Table: SGBM total time (s) for a high-dimensional problem with equal-partitioning. Test configuration: $N = 2^{20}$ and $\Delta t = T/M$.

Cont. value computation: New approach



(a) Geometric basket put option



(b) Arithmetic basket put option

Figure: CEV model convergence, $\gamma = 1.0$. Test configuration: $N = 2^{16}$, $\nu = 2^{10}$ and $d = 5$.

Cont. value computation: New approach

	Geometric basket Bermudan option			
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.000291	0.029395	0.276030	1.342147
SGBM PE	0.000274	0.029322	0.275131	1.342118
	Arithmetic basket Bermudan option			
	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1.0$
SGBM DE	0.000289	0.029089	0.267943	1.241304
SGBM PE	0.000288	0.028944	0.267214	1.225359

Table: CEV option pricing. Test configuration: $N = 2^{16}$, $\Delta t = T/4000$, $\nu = 2^{10}$ and $d = 5$.

Conclusions

- Efficient parallel GPU implementation.
- Extend the SGBM's applicability: Increasing dimensionality.
- New bundling technique.
- More general approach to compute the continuation value.
- Future work:
  - Explore the new CUDA features: i.e. cuSOLVER (QR factorization).
  - CVA calculations.

References

Shashi Jain and Cornelis W. Oosterlee.
The Stochastic Grid Bundling Method: Efficient pricing of Bermudan options and their Greeks.
Applied Mathematics and Computation, 269:412–431, 2015.

Álvaro Leitao and Cornelis W. Oosterlee.
GPU acceleration of the Stochastic Grid Bundling Method for early-exercise options.
International Journal of Computer Mathematics, 92(12):2433–2454, 2015.

Acknowledgements

MARIE CURIE ACTIONS

ITN STRIKE

This research was supported by the European Union in the FP7-PEOPLE-2012-ITN Program under Grant Agreement Number 304617 (FP7 Marie Curie Action, Project Multi-ITN STRIKE – Novel Methods in Computational Finance).

Thank you for your attention

Appendix

- Geo. basket Bermudan option - Basis functions:

$$\phi_k(\mathbf{S}_{t_m}) = \left((\prod_{\delta=1}^{d} S_{t_m}^{\delta})^{\frac{1}{d}} \right)^{k-1}, \; k = 1, \ldots, K,$$

- The expectation can directly be computed as:

$$\mathbb{E}\left[\phi_k(\mathbf{S}_{t_m}) | \mathbf{S}_{t_{m-1}}(n)\right] = \left(P_{t_{m-1}}(n) e^{\left(\bar{\mu} + \frac{(k-1)\bar{\sigma}^2}{2}\right)\Delta t} \right)^{k-1},$$

where,

$$P_{t_{m-1}}(n) = \left(\prod_{\delta=1}^{d} S_{t_{m-1}}^{\delta}(n) \right)^{\frac{1}{d}}, \; \bar{\mu} = \frac{1}{d} \sum_{\delta=1}^{d} \left(r - q_\delta - \frac{\sigma_\delta^2}{2} \right), \; \bar{\sigma}^2 = \frac{1}{d^2} \sum_{p=1}^{d} \left(\sum_{q=1}^{d} C_{pq}^2 \right)^2.$$

Appendix

- Arith. basket Bermudan option - Basis functions:

$$\phi_k(\mathbf{S}_{t_m}) = \left(\frac{1}{d}\sum_{\delta=1}^{d} S_{t_m}^{\delta}\right)^{k-1}, k = 1, \ldots, K.,$$

- The summation can be expressed as a linear combination of the products:

$$\left(\sum_{\delta=1}^{d} S_{t_m}^{\delta}\right)^{k} = \sum_{k_1+k_2+\cdots+k_d=k} \binom{k}{k_1, k_2, \ldots, k_d} \prod_{1\le\delta\le d} \left(S_{t_m}^{\delta}\right)^{k_\delta},$$

- And the expression for Geometric basket option can be applied.