NORTHWESTERN UNIVERSITY

Two-Level Simulation of Expected Shortfall: Confidence Intervals, Efficient Simulation Procedures, and High-Performance Computing

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Sciences

By

Hai Lan

EVANSTON, ILLINOIS

June 2010

ABSTRACT

Two-Level Simulation of Expected Shortfall: Confidence Intervals, Efficient Simulation Procedures, and High-Performance Computing

Hai Lan

We develop and evaluate a two-level simulation procedure that produces a confidence interval for expected shortfall. The outer level of simulation generates financial scenarios while the inner level estimates expected loss conditional on each scenario. Our procedure uses the statistical theory of empirical likelihood to construct a confidence interval, and tools from the ranking-and-selection literature to make the simulation efficient.

Parameters that govern the behavior of the simulation procedures are important to the effectiveness of sophisticated simulation. A parameter tuning method for a two-level simulation with screening is discussed. A special procedure is introduced to predict the behavior of the two-level simulation based on historical data or a pilot simulation. A hybrid method of grid search and nonlinear convex local optimization techniques is adopted to find suitable input parameters to optimize the forecast performance of the two-level simulation.

On the other hand, we accelerate the simulation by applying the new development of parallel computing into the two-level simulation with screening. A hybrid of multi-core CPU and multiple GPUs is adopted to hasten the computational efficiency of two-level simulation with screening. Currently, we can shorten the simulation time by a factor of 40 compared with serial implementation.

Acknowledgements

I would like to express my gratitude to my advisor Professors Barry L. Nelson and Jeremy Staum for their careful guidance and constant encouragement during my graduate study at Northwestern University. I would also like to thank Professor D. Apley and Professor Ming-Yang Kao for serving in my thesis committee.

I want to thank my wife Xia Zhang for her attention, understanding and continuous support for so many years. My families, no matter where they are, they are always good listeners.

Special thanks to my father. I believe you can hear me.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Financial risk management is vital to the survival of financial institutions and the stability of the financial system. A fundamental task in risk management is to measure the risk entailed by a decision, such as the choice of a portfolio. In particular, regulation requires each financial institution to measure the risk of the firm's entire portfolio and to set its capital reserves accordingly, to reduce the chance of bankruptcy if large losses occur. This firm-wide risk measurement problem is challenging. Solution methods that avoid Monte Carlo simulation involve simplifications and approximations that cast doubt on the validity of the answer. Monte Carlo simulation allows for detailed modeling of the behavior of the firm's portfolio given possible future events, and it is compatible with the use of the best available models of financial markets. Because of this, Monte Carlo simulation is an attractive methodology, but its appeal is limited by its computational cost, which can be quite large, especially when derivative securities are involved (McNeil et al., 2005, § 2.3.3). This is because a precise estimate of the risk measure requires consideration of many future financial scenarios, and it takes a long time to compute the value of all the derivative securities in any scenario. Consequently, a large firm-wide risk measurement simulation can take days to run on a cluster of one thousand computers. Because of the speed at which markets move, timelier answers are needed. One of our main contributions is to develop a more efficient simulation procedure for

risk measurement when it is time-consuming to compute the portfolio value in a future financial scenario.

Let V be a random variable representing the value of a portfolio in the future, and let F_V be its distribution. A risk measure is a functional $T(F_V)$ of this distribution. For example, value at risk VaR_{1-p} may be defined as the negative of the p-quantile of F_V. In market risk management, it is usual to consider the 95th or 99th percentile: $p = 5\%$ or 1%. In this paper, we focus on expected shortfall:

$$\mathrm{ES}_{1-p} = -\frac{1}{p}\left(\mathrm{E}[V\mathbf{1}_{\{V\le v_p\}}] + v_p(p - \Pr[V \le v_p])\right), \tag{1.1}$$

where v_p is the lower p-quantile of F_V. If F_V is continuous at v_p, then ES equals tail conditional expectation (Acerbi and Tasche, 2002):

$$\mathrm{TCE}_{1-p} = -\mathrm{E}[V|V \le v_p].$$

Closed-form expressions for ES are available when the distribution F_V belongs to some simple parametric families (McNeil et al., 2005, §§ 2.2.4, 7.2.3). There is also a literature on nonparametric estimation of expected shortfall from data $V_1, \ldots, V_k$ drawn from a stationary process whose marginal distributions are F_V. In this setting, Chen (2008) shows that although kernel smoothing is valuable in estimating VaR, the simplest nonparametric estimator of ES, involving an average of the $\lceil kp \rceil$ smallest values among $V_1, \ldots, V_k$, is preferred to kernel smoothing. Accordingly, we use unsmoothed averages in our construction of confidence intervals for ES.

However, we consider a different situation, in which we do not have a sample of data from F_V and we do not have a parametric form for F_V. In many risk measurement

applications, it is important to consider risk as depending on the current state of the market. In this case, historical loss data is not directly representative of the risks faced today. In particular, suppose that V is the gain experienced by a portfolio containing derivative securities. We have a model of underlying financial markets that allows us to sample a scenario Z (which specifies such things as tomorrow's stock prices) from its distribution F_Z, and there is a function $V(\cdot)$ such that the portfolio's gain $V = V(Z)$. Even if F_Z belongs to a simple parametric family, F_V may not, because the value function $V(\cdot)$ is not analytically tractable. Furthermore, the function $V(\cdot)$ itself is not generally known in closed form; it is known in closed form only for some simple models and derivative securities.

However, in most models, we can represent $V(Z) = \mathrm{E}[X|Z]$ where X involves the payoffs of derivative securities, which we can simulate conditional on the scenario Z.

In this situation, we can estimate the risk measure $T(F_V)$ by a two-level simulation, in which the outer level of simulation generates scenarios $Z_1, Z_2, \ldots, Z_k$ and the inner level estimates each $V_i := V(Z_i)$ by simulating V conditional on Z_i. For more on this general framework and its significance in risk management, see Lan et al. (2007b). Point estimation of a quantile of the distribution (here called F_V) of a conditional expectation via two-level simulation has been studied by Lee (1998) and Gordy and Juneja (2006, 2008). This is equivalent to point estimation of VaR. Gordy and Juneja (2008) also consider point estimation of ES. This strand of the research literature emphasizes asymptotic optimality for large computational budgets or portfolios. In related work, Steckley and Henderson (2003) study estimation of the density of F_V via two-level simulation.

This thesis focuses on interval estimation of ES and moderate sample sizes. We develop a procedure for efficient computation of a confidence interval for ES and show that it performs well at realistic sample sizes. Two-level simulations can be extremely computationally expensive. A plain two-level simulation procedure, such as that discussed in Chapter 2, can produce very wide confidence intervals given the available computational budget.

To produce a narrower confidence interval given a fixed computational budget, our procedure uses screening with common random numbers, allocates sample sizes proportional to each scenario's sample variance and computes more precise confidence intervals benefiting from the linearity of the results of empirical likelihood estimation.

Even with statistical efficiency techniques, like screening and empirical likelihood estimation, for large financial institutions with complex investment portfolios of multiple assets, the computation of market risk presents real challenges. Usually the computation of risk measures for real portfolios takes days or weeks, yet the portfolios may be refinanced within minutes or hours. This demands that computational efficiency be considered for any applicable procedure for risk measurement. Therefore, we improve the efficiency of two-level simulation in the following two ways: we automatically tune the procedure's input parameters and implement the two-level simulation on a hybrid framework of multicore CPUs and multiple Graphic Processing Units (GPUs).

For a sophisticated stochastic simulation like the two-level simulation we discuss here, the input parameters can be divided into two categories: the modeling parameters, which are part of the description of the underlying model, and the procedure parameters, which

are part of the simulation procedure. For example, the underlying stock price is a modeling parameter. The number of samples that the experimenter would like to take for each market scenario generated in the outer level to eliminate the unimportant ones is a procedure parameter. Modeling parameters are the prior knowledge the experimenters have from the calibration of raw data or theoretical analysis. Procedure parameters stand for the way the simulation is designed or the policies by which the experimental resources are allocated. Usually procedure parameters are controllable and important to the efficiency of the simulation procedure.

In practice, the procedure parameters are determined either by the experimenters' experience or from empirical rules suggested from repeated experiments with different procedure parameters. However, for these sophisticated simulation procedures, which are too expensive to run repeatedly, the tuning of procedure parameters needs special consideration. It is desirable to determine the optimal or near-optimal procedure parameters with a modest cost.

When facing an entirely new simulation problem, the experimenters might do a pilot simulation with a modest cost then tune the procedure parameters based on information from the pilot simulation. More frequently, the experimenters might face a problem similar to one seen before; for example, today's problem may be similar to yesterday's problem, but with slight changes to some of the modeling parameters. If so, the procedure parameters that were optimal for yesterday's problem should still perform very well when used in simulating today's problem. We refer to the information collected from the pilot simulation or historical simulation as pilot information.

With pilot information, we can predict the performance of the simulation with different procedure parameters. Therefore, optimization techniques may be applied to search the feasible region of the procedure parameters to optimize the forecasted performance of the simulation.

Parallel computing has proven to be effective to solve computationally intensive problems in physics, chemistry, biology and engineering. Research on parallel computing in economics and finance has been reported in the past decade, covering diverse topics from asset pricing to risk management. Yet the application of parallel computing in the financial industry is still limited as the upfront prices of huge or even middle-scale super computers are discouraging. Meanwhile, chip-level parallelism is commonplace and GPUs are fundamentally many-core processors that provide a high degree of parallelism with relatively affordable cost. Take NVIDIA GeForce 9800 as example. It contains 128 scalar processing elements per chip, and provides a friendly C interface package, called CUDA, through which programmers can take GPUs as additional computing resources where massive pieces of C codes can be run simultaneously. Naturally, the chip-level parallelism is fully compatible with distributed cluster computing and PC-grid computing (Tezuka et al. (2005)), which are economic, scalable and de facto solutions for high performance computing. Fan et al. (2004), Chinchilla et al. (2004) and Abbas-Turki and Vialle (2009) provided successful applications of the integration of chip-level parallelism with cluster computing or PC-grid computing in the fields of engineering and finance.

For our research, we only study parallel computing within one PC or workstation, which seems to be the most accessible solution for the financial industry. Currently, up to 4 GPUs can be installed within one computer. Modern CPUs usually contain 2 to

8 cores. It is reasonable to adopt OpenMP and CUDA to manipulate all computing capacities provided by CPUs and GPUs. OpenMP provides a set of pragma clauses and interfaces for developing parallel applications on multicore CPUs with shared memory.

The NVIDIA CUDA package already provides some demos on Black-Sholes pricing and Lattice pricing on a GPU. For pricing options with Levy underliers using the Fourier Space Time-stepping method, Surkov (2007) parallelized the algorithm to gain greater efficiency. As an advanced application, Singla et al. (2008) reported great speedup compared with a CPU by parallelism on a GPU or field-programmable gate arrays on the estimation of Value at Risk when the future value of the portfolio, when the future value of the portfolio, V, is directly computable.

Risk management simulations may deal with non-trivial models and thousands of derivative securities with complicated payoffs. However, for purposes of illustration, we consider the following two simple examples. This allows us to report the coverage rate that our procedure achieves by repeating the simulation experiment many times, so as to see how often our confidence interval contains the true value of ES.

1.1. Selling a Single Put Option

At time 0, we sell a put option with strike price $K = \$110$ and maturity $U = 1$ year on a stock whose initial price is $S_0 = \$100$. This stock's price obeys the Black-Scholes model with drift $\mu = 6\%$ and volatility $\sigma = 15\%$. There is a money market account with interest rate $r = 6\%$. The initial price for which we sell the put option is $P_0 = P(U, S_0)$, which is the Black-Scholes formula evaluated for maturity U and stock price S_0.

We are interested in $\mathrm{ES}_{0.99}$ at time $T = 1/52$ years, or one week from now. The scenario Z is a standard normal random variable that determines the stock price at time T:

$$S_T = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) T + \sigma\sqrt{T} Z\right).$$

The net payoff at maturity U, discounted to time T, from selling the put for an initial price of P_0 is

$$X = e^{-r(U-T)} \left(P_0 e^{rU} - (K - S_U)^+\right),$$

where

$$S_U = S_T \exp\left(\left(r - \frac{\sigma^2}{2}\right)(U - T) + \sigma\sqrt{U - T} Z'\right)$$

and Z' is a standard normal random variable independent of Z.

In this simple example, we can actually find the value

$$V = \mathrm{E}[X|Z] = P_0 e^{rT} - P(U - T, S_T),$$

using the Black-Scholes formula evaluated for maturity $U - T$ and stock price S_T. Furthermore, V is strictly decreasing in Z, so we can compute that $\mathrm{VaR}_{0.99} \approx \$2.92$ by evaluating V at $Z = z_{0.01}$, the standard normal first percentile. By numerical integration, we can also compute $\mathrm{ES}_{0.99} \approx \$3.39$, which will help us to evaluate the performance of our procedure. (Our procedure does not compute V by using the Black-Scholes formula, but rather estimates it using inner-level simulation of payoffs at maturity.)

Table 1.1. Portfolio of Call Options

Index i	Underlying Stock	Position θ_i	Strike K_i	Maturity U_i	Option Price	Implied Volatility σ_i
1	CSCO	200	\$27.5	0.315	\$1.65	26.66%
2	CSCO	-400	\$30	0.315	\$0.7	25.64%
3	CSCO	200	\$27.5	0.564	\$2.5	28.36%
4	CSCO	-200	\$30	0.564	\$1.4	26.91%
5	JAVA	600	\$5	0.315	\$0.435	35.19%
6	JAVA	1200	\$6	0.315	\$0.125	35.67%
7	JAVA	-900	\$5	0.564	\$0.615	36.42%
8	JAVA	-300	\$6	0.564	\$0.26	35.94%

1.2. A Portfolio of Options on Two Stocks

We are interested in ES at time $T = 1/365$ years, or one day, of a portfolio of call options on Cisco (CSCO) and Sun Microsystems (JAVA), as shown in Table 1.1. In the table, the position given for each option is the number of shares of stock we have the option to buy; if it is negative, then our portfolio is short call options on that many shares of stock. Except for the portfolio weights, which we made up, the data in the table was drawn from listed options prices on June 26, 2007. We ignored the distinction between American and European options because neither of these stocks pays dividends, a situation in which early exercise of an American call option is widely regarded as mistaken (see, e.g., Luenberger, 1998, § 12.4).

The scenario $Z = (Z^{(1)}, Z^{(2)})$ is a bivariate normal random variable that determines the stock prices at time T:

$$S_T^{(j)} = S_0^{(j)} \exp\left(\left(\mu^{(j)} - \frac{1}{2}(\sigma^{(j)})^2\right) T + \sigma^{(j)}\sqrt{T}Z^{(j)}\right), j = 1, 2.$$

Based on sample moments of 1,000 daily stock prices, the volatilities of CSCO and JAVA are respectively $\sigma^{(1)} = 32.85\%$ and $\sigma^{(2)} = 47.75\%$, while the correlation between the components of Z is 0.382. In practice, more sophisticated methods of volatility forecasting would be used, but this method yields a reasonable covariance matrix for the vector S_T of stock prices tomorrow. Because one day is such a short period of time that the effect of the drift μ is negligible, while mean returns are hard to estimate because of the high ratio of volatility to mean, we take each $\mu^{(j)} = 0$.

In addition to a distribution F_Z for scenarios, we need to specify the function $V(\cdot)$ by saying how option values at time T depend on the scenario. We adopt the "sticky strike" assumption, according to which each option's value at time T is given by the Black-Scholes formula with volatility equal to the implied volatility that this option had at time 0 (Derman, 1999). This does not make for an arbitrage-free model of the underlying stock prices S, but it is an assumption that has been used in practice to model short-term changes in option values. As in the previous example, we can compute these values without using inner-level simulation, but our procedure performs inner-level simulation for each option i by taking the stock price at maturity U_i to be

$$S_i = \frac{S_T^{(j_i)}}{D_i} \exp\left(-\frac{1}{2}\sigma_i^2(U_i - T) + \sigma_i\sqrt{U_i - T}Z_i'\right)$$

where $j_1 = j_2 = j_3 = j_4 = 1$ (the four options on CSCO) and $j_5 = j_6 = j_7 = j_8 = 2$ (the four options on JAVA), D_i is a discount factor from T to U_i, and Z' is a standard multivariate normal random vector independent of Z. Based on Treasury bond yields, the discount factor was 0.985 for options maturing in 0.315 years and 0.972 for options maturing in 0.564 years. The independence of the components of Z' means that, even

though in reality the eight options depend on two correlated stock prices at two times, independent inner-level simulations are used to estimate the option prices at time T by default. As shown by Gordy and Juneja (2006, 2008), this can improve the efficiency of the two-level simulation. Furthermore, the sticky strike assumption does not lead to a consistent model of the underlying stock prices, so one can not use a single simulation of the two stocks to price all the options; this makes it more natural to think of eight separate option pricing simulations.

The value of option i at time T is the conditional expectation of the discounted payoff $Y_i := D_i(S_i - K_i)^+$ given $S_T^{(j_i)}$. The profit from holding the portfolio from 0 to T is $V(Z) = \mathrm{E}[X|Z]$ where $X = \theta^\top(Y - P_0/D_0)$ and the discount factor $D_0 \approx 1$ because the time value of money over one day is negligible; $\theta^\top$ is the vector standing for the positions of all instruments held in the invested portfolio. We estimated the true value of $\mathrm{ES}_{0.99}$ to be \$32.4, the average point estimate produced by 100 repetitions of the complete experiment with a budget of 1.56 billion inner-level simulations each.

1.2.1. Different Pricing Models: Tailored by CRN

The function of a pricing model is to set up a map between the scenarios and portfolio's values. In most cases, this map has no closed form or is difficult to calculate directly. Monte Carlo simulation is adopted to estimate the portfolio's value conditionally on each market scenario by averaging an amount of payoffs in the risk neutral probability of each instrument in the portfolio. Whether the simulation of payoffs of different instruments is independent or not has nothing to do with the validity of the two-level simulation.[1]

[1]This is different from the simulation of the price of a basket option, where the dependence of the prices of underlying asserts DOES affect the price of the option.

Yet smaller variance of the simulated portfolio value helps to generate a more precise confidence interval for the expected shortfall.

Take the portfolio of options on JAVA and CISCO as an example. We could apply different random number generating strategies shown in Table 1.2.

Option	CRN1	CRN2	NATURE	NAIVE
C1	Z_1	Z	Z'_1	Z_1
C2	Z_1	Z	Z'_2	Z_2
C3	$-Z_1$	$-Z$	Z'_3	Z_3
C4	Z_1	$-Z$	Z'_4	Z_4
C5	Z_2	Z	Z'_5	Z_5
C6	Z_2	$-Z$	Z'_6	Z_6
C7	Z_2	Z	Z'_7	Z_7
C8	Z_2	Z	Z'_8	Z_8

Table 1.2. Design of Pricing Model

In Table 1.2, Z is a standard normal random variable, $Z_i, i = 1, \ldots, 8$ are independent standard normal random variables and Z'_i are standard normal distributed random variables with relationships shown as follows:

$$Z'_i = Z'_{i+1} \quad , \quad i = 1, 3, 5, 7$$

$$\rho(Z'_1, Z'_3) = 0.747$$

$$\rho(Z'_1, Z'_4) = 0.382$$

$$\rho(Z'_5, Z'_7) = 0.747$$

Strategy CRN1 and CRN2 are just two different configurations of Common Random Number (CRN; see, e.g., Law and Kelton, 2000) applied in the pricing model. When simulating the value of portfolio's with the risk neutral probability, the correlation implied

in the real world probability is introduced to reduce variance. This strategy is called NATURE. Strategy NAIVE, the default strategy adopted, simply summarizes all the weighted, independently simulated value of each instrument altogether.

Through this approach, we enrich the test examples without introducing any more portfolios. The topic about how to design optimal random numbers to reduce the variance is out of the scope of this thesis.

CHAPTER 2

Plain Two-Level Simulation

The plain two-level simulation addressed here works as the starting point of our research and also as a benchmark to compare to in the empirical experiments. The mechanism of confidence interval estimation on two-level simulation is based on the fact that $[\inf_{\mathbf{v}\in\mathcal{V}} G(\mathbf{v}), \sup_{\mathbf{v}\in\mathcal{V}} G(\mathbf{v})]$ is a confidence interval for $G(\mathbf{V})$ with confidence level at least $1-\alpha_i$ if $\mathcal{V}$ is a $1-\alpha_i$ confidence region for $\mathbf{V}$. Note that G could be any function, yet computation of confidence interval becomes extremely easy when G is monotone or even linear.

2.1. The Plain Procedure

Two-level simulation has wide application in discrete-event simulation, such as simulation of queueing systems or stochastic optimization problems, where the two levels have different meanings. Here we only focus on its application in risk management in financial markets. Market scenarios, $Z_i, i = 1, \ldots,$, are generated in the outer level; Z_i contains financial variables such as stock prices or interest rates at a future date T. Conditional portfolio values, $V_i := V(Z_i) = \mathbf{E}[V|Z_i]$, are computed in the inner level. When V_i's have to be estimated by Monte Carlo simulation, payoffs at (or until) some future time U_j (where j is the index of the instruments held in the portfolio) need to be sampled. During the sampling, the financial variables within market scenarios may need to be simulated forward between $[T, U_j]$ conditionally on Z_i. Since valuing the instruments' conditional

prices is the goal of inner-level simulation, these financial variables usually modeled as having so-called "risk natural probability" which is an artificial probability measure to express the value of market instruments as the expectation of future payoffs.

Figure 2.1 shows the architecture of two-level simulation. Suppose the total number of simulated payoffs C is given by the experimenter as a measure of the available computational resources, and the number of market scenarios k is also specified. Without prior information about each market scenario, the sample size of the conditional payoffs of each market scenario, N_i has to be set to equal, i.e. $N_i = n_0 = \frac{C}{k}$. There are some parts of the plain two-level simulation to which efficiency techniques can be applied. Firstly, for ES only a small portion of the market scenarios are important to our calculation. A large proportion of total computational resources are wasted if equal simulation effort on all generated market scenarios. Secondly, even for the market scenarios which are important to the calculation of ES, there are still differences in the sample variance of $X_{i(\cdot)}$ which is an unbiased estimator of V_i. If information about the market scenarios can be collected through early stage simulation in a multi-stage procedure, an optimal budget allocation of N_i is preferable.

2.2. Confidence Interval for Plain Two-Level Simulation

How does statistical uncertainty at the inner level combine with statistical uncertainty at the outer level? This question must be answered to create a confidence interval after a two-level simulation. One sign of the difficulties is that uncertainty at the inner level can lead to bias in the obvious point estimates of $\mathrm{ES}_{1-p}(F_V)$. Then the simple idea to construct a confidence interval by applying the Center Limit Theorem to the average of the

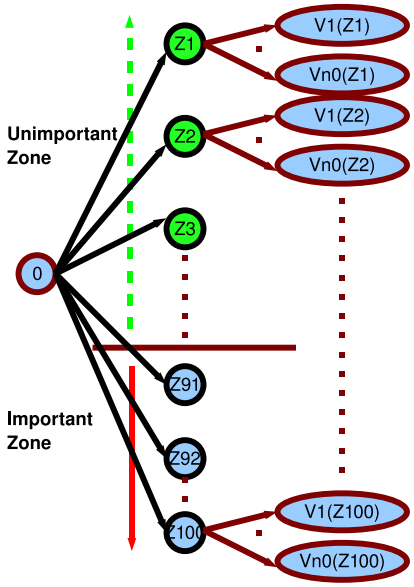

Figure 2.1. Plain Two-Level Simulation

point estimators does not apply. The problem can be viewed as follows. If V_i and its non-decreasing order $\pi_V(\cdot)$ were known, a confidence interval $[L(\mathbf{V}),\ U(\mathbf{V})]$ for $\mathrm{ES}_{1-p}(F_V)$ can be computed from $\mathbf{V} = (V_1, \cdots, V_k)$ either from a parametric or nonparametric method. We relay on empirical likelihood estimator here because very weak assumptions are required for this nonparametric method. Baysal and Staum (2008) studied the construction of outer-level confidence for $\mathrm{ES}_{1-p}(F_V)$ based on the empirical likelihood estimator. Here we review some essential facts about empirical likelihood for understanding its application in our two-level simulation. Empirical likelihood involves considering distributions that arise by assigning a vector $\mathbf{w}$ of weights to the scenarios $Z_1, Z_2, \ldots, Z_k$, or, equivalently,

to their values $V_1, V_2, \ldots, V_k$. This weight vector $\mathbf{w}$ must belong to the set

(2.1)
$$\mathcal{S}(k) := \bigcup_{\ell=1}^{k} \mathcal{S}_\ell(k) \text{ where } \mathcal{S}_\ell(k) := \left\{ \mathbf{w} : \mathbf{w} \geq 0,\ \sum_{i=1}^{k} w_i = 1,\ \sum_{i=1}^{\ell} w_i = p,\ \prod_{i=1}^{k} w_i \geq c\, k^{-k} \right\},$$

where c is a critical value derived from a chi-squared distribution. Each $\mathbf{w} \in \mathcal{S}(k)$ belongs to $\mathcal{S}_\ell(k)$ for a unique integer ℓ, which can be interpreted as the number of scenarios that we believe belong to the tail of F_V, i.e. are less than or equal to the p-quantile of V. The intuition behind using only weight vectors that fall in $\mathcal{S}(k)$ in this application of empirical likelihood is as follows. First, it is unlikely that too few or too many of the scenarios that we sampled from F_V belong to the tail of F_V. There are integers $\ell_{\min}$ and $\ell_{\max}$ such that $\mathcal{S}_\ell(k)$ is empty if $\ell < \ell_{\min}$ or $\ell > \ell_{\max}$; we need only consider a limited range of ℓ, not all $1, 2, \ldots, k$. (Although ℓ depends on $\mathbf{w}$, and $\ell_{\min}$ and $\ell_{\max}$ depend on k, to lighten notation we do not make this dependence explicit.) Second, we use only weight vectors that are fairly close to uniform; if elements w_i are too far from $1/k$, then $\prod_{i=1}^{k} w_i < c\, k^{-k}$. This means that we work with discrete distributions that are not too far from the empirical distribution which places weight $1/k$ on each of $V_1, V_2, \ldots, V_k$.

Because ES_{1-p} involves an average over the left tail containing probability p, we also define a transformed weight vector $\mathbf{w}'$:

(2.2)
$$w_i' := \begin{cases} -w_i/p, & i = 1, 2, \ldots, \ell \\ 0, & \text{otherwise.} \end{cases}$$

If the vector $\mathbf{V} := (V_1, V_2, \ldots, V_k)$ of true portfolio values were known, then with a weight vector $\mathbf{w}$ it would define a discrete distribution $F_{\mathbf{w},\mathbf{V}}$ assigning probability w_i to each value V_i. For this distribution, ES_{1-p} is $\sum_{i=1}^{k} w_i' V_{\pi_V(i)}$. The empirical likelihood confidence

interval for ES_{1-p} of the unknown true distribution F_V, expressed in Equation (1.1), is

$$\left[L(\mathbf{V}) = \min_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' V_{\pi_V(i)},\ U(\mathbf{V}) = \max_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' V_{\pi_V(i)}\right], \tag{2.3}$$

representing the outer-level uncertainty entailed by working with a sample $Z_1, Z_2, \ldots, Z_k$ instead of the true distribution F_V and satisfying

$$\lim_{k\to\infty} \Pr\{\mathrm{ES}_{1-p}(F_V) \in [L(\mathbf{V}),\ U(\mathbf{V})]\} \geq 1 - \alpha_o. \tag{2.4}$$

The intuition behind (2.3) is that $\mathcal{S}(k)$ is the set of weight vectors that are "empirically likely," and we do not believe that the true ES according to F_V is less than the smallest ES that comes from applying weights in $\mathcal{S}(k)$ to the values $\mathbf{V}$, nor do we believe that the true ES is more than the largest ES that comes from applying weights in $\mathcal{S}(k)$ to the values $\mathbf{V}$.

Because V_i is not known, $\mathbf{V}$ and hence $L(\mathbf{V})$ and $U(\mathbf{V})$ are unobservable. Instead we construct $\hat{L}$ and $\hat{U}$ from the observable sample of payoffs to satisfy, for all $\mathbf{V}$,

$$\lim_{\mathbf{N}\to\infty} \Pr\{[L(\mathbf{V}), U(\mathbf{V})] \subseteq [\hat{L},\ \hat{U}]\} \geq 1 - \alpha_i \tag{2.5}$$

where $\mathbf{N} = (N_1, \cdots, N_k)$. Applying the Bonferroni inequality to Inequalities (2.4) and (2.5),

$$\lim_{k,\mathbf{N}\to\infty} \Pr\{\mathrm{ES}_{1-p}(F_V) \in [\hat{L},\ \hat{U}]\} \geq 1 - \alpha, \tag{2.6}$$

where $\alpha = \alpha_i + \alpha_o$, an error spending decomposition which expresses the fact that error arises in one of two ways: the unknown outer-level confidence interval $[L(\mathbf{V}),\ U(\mathbf{V})]$ fails

to cover the true value due to unlucky sampling of risk factors, or the known confidence interval $[\hat{L},\ \hat{U}]$ fails to cover the unknown outer-level confidence interval due to unlucky sampling of payoffs at the inner level.

Suppose that sampling is done so the risk factors $Z_1, \cdots, Z_k$ are independent and X_{i_1,j_1}, X_{i_2,j_2} are independent of each other conditional on Z_{i_1} and Z_{i_2} for $i_1 \neq i_2$. As $N_i \to \infty$, $\sqrt{N_i}\bar{X}_i(N_i)$ converges to normal distribution and are mutually independent. Define $\epsilon := 1 - (1-\alpha_i)^{1/k}$ and let t_i be the quantile at the $1-\epsilon/2$ level of the Student t distribution with $N_i - 1$ degrees of freedom. Let $\mathcal{V}$ be the random k-dimensional box formed as the Cartesian product over $i = 1, \ldots, k$ of the intervals

$$\left[\bar{X}_i(N_i) - t_i S_i/\sqrt{N_i}, \bar{X}_i(N_i) + t_i S_i/\sqrt{N_i}\right].$$

Then $\mathcal{V}$ is an asymptotically valid confidence region for $\mathbf{V}$:

$$\lim_{\mathbf{N}\to\infty} \Pr\{\mathbf{V} \in \mathcal{V}\} \geq (1-\epsilon)^k = 1 - \alpha_i.$$

Consequently, Inequality (2.5) is satisfied with

$$\hat{L} = \inf_{v\in\mathcal{V}} L(v) \text{ and } \hat{U} = \sup_{v\in\mathcal{V}} U(v). \tag{2.7}$$

It would also be possible to construct $\mathcal{V}$ in different ways, e.g., as an ellipsoid based on the limiting χ^2_k distribution of $\sum_{i=1}^k (\bar{X}_i(N_i) - V_i)^2/S_i^2 N_i$. The ease of performing the optimizations in Equation (2.7) and the size of the resulting confidence interval are relevant considerations in choosing how to construct $\mathcal{V}$. These optimizations are trivial as L and U are linear when empirical likelihood estimation applied in the outer level and

$\mathcal{V}$ has a least and a greatest element with respect to the usual partial ordering on k , as it does when defined as the k-dimensional box.

Another important inequality can help to estimate $\hat{U}$ more precisely,

$$\max_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' \bar{X}_{\pi_V(i)}(N_{\pi_V(i)}) \leq \max_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1,\ldots,k} w_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) \tag{2.8}$$

where $\pi_1(\cdot)$ is the permutation of $i = 1, \ldots, k$ such that $\bar{X}_{\pi_1(i)}(N_{\pi_1(i)})$ is nondecreasing in i, while π_V is another permutation such that $V_{\pi_V(1)} \leq V_{\pi_V(2)} \leq \cdots \leq V_{\pi_V(k)}$. Consequently, $\hat{U}$ can be refined as

$$\hat{U} = \max_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) + z_{\text{hi}} \max_{i=1,\ldots,k} S_i \sqrt{\max_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i'^2} \tag{2.9}$$

where z_{hi} is the $1 - \alpha_{hi}$ quantile of the standard normal distribution. Also $\hat{L}$ defined in Equation (2.7) can be explicitly expressed as

$$\hat{L} = \min_{\mathbf{w}\in\mathcal{S}} \sum_{i=1}^{\ell_{\max}} w_i' \left(\bar{X}_{\pi_{Li}}(N_{\pi_{Li}}) + z_{\text{lo}} \frac{S_{\pi_{Li}}(N_{\pi_{Li}})}{\sqrt{N_{\pi_{Li}}}} \right) \tag{2.10}$$

where π_L satisfies that $\bar{X}_{\pi_{Li}}(N_{\pi_{Li}}) + z_{lo} S_{\pi_{Li}}(N_{\pi_{Li}})/\sqrt{N_{\pi_{Li}}}$ is nondecreasing in i, and z_{lo} is the $1 - (1 - \alpha_{lo})^{1/k}$ quantile of the standard normal distribution.

CHAPTER 3

Two-Level Simulation with Screening

Efficiency is important for two-level simulation, which can be very computationally expensive, as discussed in Chapter 2. Efficiency enhancement for two-level simulation is interesting for several reasons. It may not be straightforward to get a confidence interval by the methods described in Chapter 2.2 when using variance reduction. The shape of the inner-level confidence region $\mathcal{V}$ greatly influences efficiency. There are questions about how to allocate computational resources within the inner level and how to allocate computational resources between the outer and inner levels. The latter issue is addressed in Chapter 4.

3.1. Screening

Screening is the process of eliminating ("screening out") scenarios to increase the simulation's efficiency by devoting more computational resources to the remaining scenarios. From Equation (1.1) in Chapter 1, we can see that the point estimation of ES depends on the values of scenarios $\pi_V(1), \pi_V(2), \ldots, \pi_V(\lceil kp \rceil)$ alone, so we want screening to keep these scenarios but eliminate as many others as possible. Call the set of scenarios that survive screening I, and define $\gamma := \{\pi_V(1), \pi_V(2), \ldots, \pi_V(\lceil kp \rceil)\}$, the set of scenarios we wish to keep. The event of correct screening is $\{\gamma \subseteq I\}$, and we must create a screening procedure such that $\Pr\{\gamma \subseteq I\} \geq 1 - \alpha_{\mathrm{s}}$. The number of pairwise comparisons between γ and all other scenarios is $(k - \lceil kp \rceil)\lceil kp \rceil$. Therefore, for each ordered pair (i, j) we

consider a hypothesis test that $V_i \leq V_j$ at level $\alpha_s/((k - \lceil kp \rceil)\lceil kp \rceil)$. If the hypothesis is rejected, then we say Z_i is "beaten" by Z_j. For each $i = 1, 2, \ldots, k$, let $X_{i1}, X_{i2}, \ldots, X_{in_0}$ be an i.i.d. sample drawn from the conditional distribution of X given Z_i, and let $\bar{X}_i(n_0)$ be its sample average. For each $i, j = 1, 2, \ldots, k$, let $S_{ij}^2(n_0)$ be the sample variance of $X_{i1} - X_{j1}, X_{i2} - X_{j2}, \ldots, X_{in_0} - X_{jn_0}$. We retain all scenarios that are beaten fewer than $\lceil kp \rceil$ times:

$$I = \left\{ i : \sum_{i \neq j} \mathbf{1}\left\{ \bar{X}_i(n_0) > \bar{X}_j(n_0) + d\frac{S_{ij}(n_0)}{\sqrt{n_0}} \right\} < \lceil kp \rceil \right\} \tag{3.1}$$

where $\mathbf{1}\{\cdot\}$ is an indicator function and

$$d = t_{n_0 - 1, 1 - \alpha_s/((k - \lceil kp \rceil)\lceil kp \rceil)} \tag{3.2}$$

is the $1 - \alpha_s/((k - \lceil kp \rceil)\lceil kp \rceil)$ quantile of the t-distribution with $n_0 - 1$ degrees of freedom. Because of the extra margin $dS_{ij}(n_0)/\sqrt{n_0}$ in Equation (3.1), there are pairs (i, j) such that neither does scenario i beat scenario j, nor does scenario j beat scenario i; therefore it is possible for $|I| > \lceil kp \rceil$ scenarios to survive screening.

From Equation (3.1) we see that it is easier to screen out scenarios when the sample variances $S_{ij}^2(n_0)$ are smaller. We use CRN to reduce the variance of $X_i - X_j$ by inducing positive correlation between X_i and X_j. In finance, CRN usually induces a substantial positive correlation between X_i and X_j. In our examples, X_i and X_j have to do with stock option payoffs at maturity $U > T$ simulated conditional on the stock price at time T being $S_i(T)$ or $S_j(T)$ respectively (here all instruments have the same maturity). These payoffs

are highly correlated when CRN is used, because, in our examples, CRN makes the stock return between times T and U the same for any value of $S(T)$.

3.2. Explanation of the Procedure

This section presents a fixed-budget two-level simulation procedure for generating a confidence interval for ES_{1-p}. The procedure first simulates scenarios $Z_1, Z_2, \ldots, Z_k$. If the values $V_1, V_2, \ldots, V_k$ of these scenarios were known, then the point estimate of ES_{1-p} would be

$$-\frac{1}{p}\left(\sum_{i=1}^{\lfloor kp \rfloor}\frac{1}{k}V_{\pi_V(i)} + \left(p - \frac{\lfloor kp \rfloor}{k}\right)V_{\pi_V(\lceil kp \rceil)}\right). \tag{3.3}$$

That is, $V_{\pi_V(i)}$ is the ith order statistic of $V_1, V_2, \ldots, V_k$.

Because these values are not known, they are estimated by inner-level simulation. The inner level of simulation has a first stage in which $n_0 \geq 2$ payoffs are generated for every scenario, using common random numbers. Our proof of the procedure's validity depends on an assumption that the payoffs are normally distributed. Financial payoffs are often far from normally distributed, but the sample averages are usually close to normally distributed because of the Central Limit Theorem. Lesnevski et al. (2008) investigated empirically the effect of non-normal payoffs on a related confidence interval procedure and found that it posed no problem as long as the first-stage sample size n_0 was at least 30.

After the first stage, screening eliminates scenarios whose values are not likely to appear in Equation (3.3). Then sample sizes $N_1, N_2, \ldots, N_k$ are chosen; the sample size N_i is 0 if scenario Z_i has been screened out. The goal of screening is to allocate more of the computational budget to scenarios that matter. Next, the first-stage data are discarded, a

process called "restarting." This is necessary for the statistical validity of the confidence interval (Boesel et al., 2003).

In the second stage, N_i payoffs $X_{i1}, X_{i2}, \ldots, X_{iN_i}$ are generated conditional on the scenario Z_i for each $i = 1, 2, \ldots, k$ using independent sampling (no CRN). The sample average of $X_{i1}, X_{i2}, \ldots, X_{iN_i}$ is denoted $\bar{X}_i(N_i)$. Then a confidence interval is formed: the confidence limits appear in Equations (3.4) and (3.5) below. The result derived in Appendix B, is that the lower confidence limit is

$$\min_{\ell=\lfloor kp \rfloor,\ldots,\ell_{\max}} \left(\min_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\text{lo}}(\ell) B_0(\ell) \right) \tag{3.4}$$

and the upper confidence limit is

$$\max_{\ell=\ell_{\min},\ldots,\lceil kp \rceil} \left(\max_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) + z_{\text{hi}} B_S(\ell) \right), \tag{3.5}$$

where several quantities are defined in Step 5 of the procedure in the following subsection. At an intuitive level, the lower confidence limit in Equation (3.4) arises from Equation (2.3) by ordering the scenarios based on information available at the end of the first stage, estimating the scenarios' values by second-stage sample averages, and subtracting a term that accounts for inner-level uncertainty. The upper confidence limit in Equation (3.5) arises similarly, but the ordering of the scenarios is based on information available at the end of the second stage, and we add a different term to account for inner-level uncertainty. The minimization and maximization over ℓ represent our uncertainty about how many of the values $V_1, V_2, \ldots, V_k$ are less than the quantile v_p.

The two-level simulation point estimate of ES_{1-p} is

$$-\frac{1}{p}\left(\sum_{i=1}^{\lfloor kp\rfloor}\frac{1}{k}\bar{X}_{\pi_1(i)}(N_{\pi_1(i)})+\left(p-\frac{\lfloor kp\rfloor}{k}\right)\bar{X}_{\pi_1(\lceil kp\rceil)}(N_{\pi_1(i)})\right). \tag{3.6}$$

If $N_i = 0$, then $\bar{X}_i$ is taken to be ∞ so that it is not among the order statistics used in Equation (3.6).

To get a confidence interval, we need a way of combining uncertainty that arises at the outer level, because $Z_1, Z_2, \ldots, Z_k$ is a sample from F_Z, with uncertainty that arises at the inner level because we only possess an estimate $\bar{X}_i(N_i)$ of each scenario's value $V_i = V(Z_i)$. In Chapter 2, we described a framework for two-level simulation that generates a two-sided confidence interval $[\hat{L}, \hat{U}]$ with confidence level $1-\alpha$ where α can be decomposed as $\alpha = \alpha_{\mathrm{o}} + \alpha_{\mathrm{i}}$, representing errors due to the outer and inner levels of simulation, respectively. Here we further decompose $\alpha_{\mathrm{i}} = \alpha_{\mathrm{s}} + \alpha_{\mathrm{hi}} + \alpha_{\mathrm{lo}}$, where α_{s} is error due to screening and α_{hi} and α_{lo} are errors respectively associated with upper and lower confidence limits for inner-level simulation.

3.2.1. Outline of the Procedure

The procedure involves a fixed computational budget, which may be expressed as a total number C of simulation replications, i.e., the total number of payoffs that can be simulated, or as an amount of computing time T. The distinction between these two kinds of computational budgets is important when choosing the first-stage sample size n_0 and the number of scenarios k. However, given n_0 and k, the kind of budget makes only a small difference in determining the number C_1 of payoffs to simulate in the second stage. If the budget is C total payoffs, then $C_1 = C - kn_0$. If the budget is an amount of

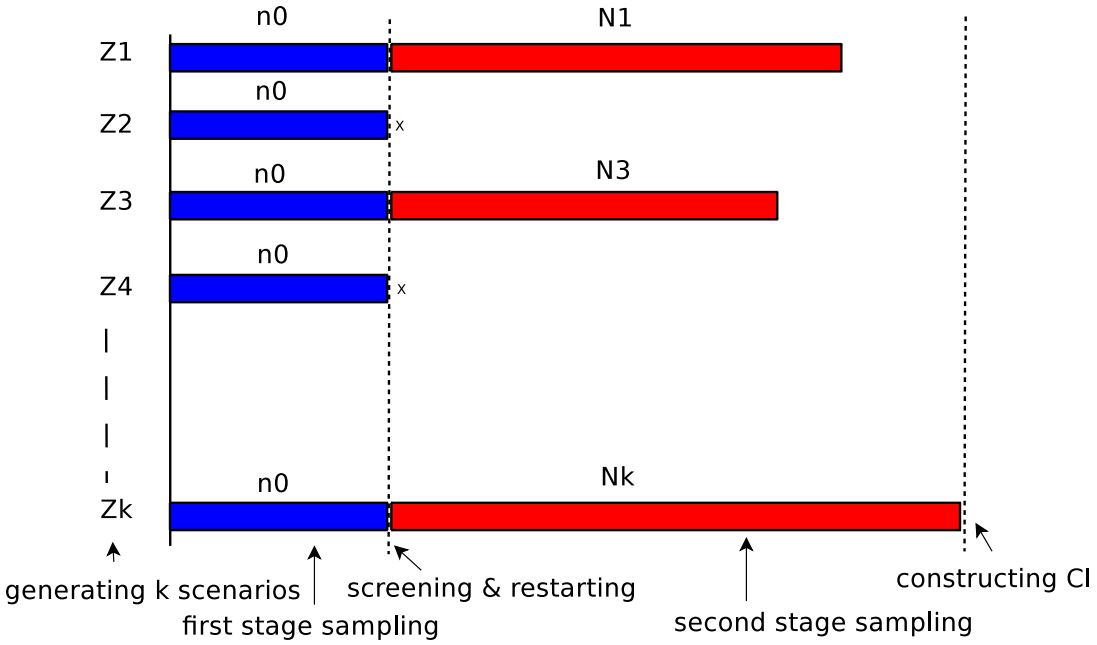

Figure 3.1. Schematic illustration of our procedure's operation.

time T, then during the first stage we must estimate t, the amount of time required to simulate one payoff, and record T_0, the amount of time required by the first stage. Then $C_1 = (T - T_0)/t$, treating the amount of time required to construct the confidence interval at the end as negligible in comparison to simulating payoffs.

To explain exactly how CRN are used, we overload notation by supposing that there is a function $X_i(\cdot)$ such that when U is a uniform random variate (or vector), the distribution of $X_i(U)$ is the conditional distribution of the payoff X given that the scenario is Z_i.

The procedure has the following steps, illustrated in Figure 3.1:

(1) **Scenario Generation**:

Generate scenarios $Z_1, Z_2, \ldots, Z_k$ independently from the distribution F_Z.

(2) **First Stage Sampling**:

Sample $U_1, U_2, \ldots, U_{n_0}$ independently from a uniform distribution.

For each $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, n_0$, compute $X_{ij} := X_i(U_j)$.

(3) **Screening**:

For each $i = 1, 2, \ldots, k$, compute the sample average $\bar{X}_i(n_0)$ and sample variance $S_i^2(n_0)$ of $X_{i1}, X_{i2}, \ldots, X_{in_0}$.

Sort to produce a permutation π_0 of $\{1, 2, \ldots, k\}$ such that $\bar{X}_{\pi_0(i)}(n_0)$ is nondecreasing in i.

Compute $\tilde{S}^2(n_0) := \max_{i=1,2,\ldots,\lceil kp \rceil} S^2_{\pi_0(i)}$ and d according to Equation (3.2).

Initialize $I \leftarrow \{1, 2, \ldots, \lceil kp \rceil\}$ and $i \leftarrow k$.

(a) *Screening of scenario $\pi_0(i)$:* Initialize $b \leftarrow 0$ and $j \leftarrow 1$.

(i) Compute the sample variance $S^2_{\pi_0(i)\pi_0(j)}$ of $X_{\pi_0(i)1} - X_{\pi_0(j)1}, X_{\pi_0(i)2} - X_{\pi_0(j)2}, \ldots, X_{\pi_0(i)n_0} - X_{\pi_0(j)n_0}$.

(ii) If $\bar{X}_{\pi_0(i)}(n_0) > \bar{X}_{\pi_0(j)}(n_0) + dS_{\pi_0(i)\pi_0(j)}/\sqrt{n_0}$, scenario $\pi_0(i)$ beats scenario $\pi_0(j)$: set $b \leftarrow b + 1$.

(iii) If $b \geq \lceil kp \rceil$, scenario $\pi_0(i)$ is screened out: go to Step (3b). Otherwise, set $j \leftarrow j + 1$.

(iv) If $j < i$, go to Step 3(a)i. Otherwise, scenario $\pi_0(i)$ survives screening: set $I \leftarrow I \cup \{\pi_0(i)\}$.

(b) *Loop:* Set $i \leftarrow i - 1$. If $i > \lceil kp \rceil$, go to Step (3a).

(4) **Restarting and Second Stage Sampling**:

Discard all payoffs from Step 2.

Compute $C_1 := C - kn_0$ or $(T - T_0)/t$, depending on the type of budget constraint.

For each $i \in I$, compute

$$N_i := \left\lceil \frac{C_1 S_i^2(n_0)}{\sum_{j \in I} S_j^2(n_0)} \right\rceil . \tag{3.7}$$

For each $i \in I$ and $j = 1, 2, \ldots, N_i$, sample U_{ij} independently from a uniform distribution and compute $X_{ij} := X_i(U_{ij})$.

(5) **Constructing the Confidence Interval**:

For each $i \in I$, compute the sample average $\bar{X}_i(N_i)$ and sample variance $S_i^2(N_i)$ of $X_{i1}, X_{i2}, \ldots, X_{iN_i}$, and compute $s_i := \sqrt{S_i^2(N_i)/N_i}$.

Compute

$$\begin{aligned} \ell_{\min} &:= \min \left\{ \ell : k^k \left(\frac{p}{\ell}\right)^{\ell} \left(\frac{1-p}{k-\ell}\right)^{k-\ell} \geq c \right\} \quad \text{and} \\ \ell_{\max} &:= \max \left\{ \ell : k^k \left(\frac{p}{\ell}\right)^{\ell} \left(\frac{1-p}{k-\ell}\right)^{k-\ell} \geq c \right\}, \end{aligned}$$

respectively, the smallest and largest numbers of scenarios to use in estimating ES, where c is the threshold in the empirical likelihood estimator.

Initialize the lower confidence limit $\hat{L} \leftarrow \infty$. Compute $N_{\text{lo}}(\lceil kp \rceil) := \min_{i=1,2,\ldots,\lceil kp \rceil} N_{\pi_0(i)}$ and $\underline{s}(\lceil kp \rceil) := \max_{i=1,2,\ldots,\lceil kp \rceil} s_{\pi_0(i)}$, which are respectively the smallest sample size and the largest standard error associated with any of the $\lceil kp \rceil$ scenarios with the lowest first-stage sample averages.

The following loop computes the lower confidence limit as a minimum of lower bounds associated with different numbers ℓ of scenarios that could be used in estimating ES. For $\ell = \lfloor kp \rfloor, \lfloor kp \rfloor + 1, \ldots, \ell_{\max}$,

(a) Compute $z_{\text{lo}}(\ell) := t_{1-\alpha_{\text{lo}},N_{\text{lo}}(\ell)-1}$,

$$\Delta(\ell) := \sqrt{\max_{\mathbf{w}\in\mathcal{S}(\ell)} \sum_{i=1}^{\ell} (w_i')^2}, \tag{3.8}$$

and $B_0(\ell) := \underline{s}(\ell)\Delta(\ell)$, which serves to bound standard error in estimating ES using ℓ scenarios.

(b) Set

$$\hat{L} \leftarrow \min\left\{\hat{L}, \min_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\text{lo}}(\ell) B_0(\ell)\right\}. \tag{3.9}$$

(c) Compute $N_{\text{lo}}(\ell{+}1) := \min\{N_{\text{lo}}(\ell), N_{\pi_0(\ell+1)}\}$ and $\underline{s}(\ell{+}1) \leftarrow \max\{\underline{s}(\ell), s_{\pi_0(\ell+1)}\}$.

Sort to produce a mapping π_1 from $\{1, 2, \ldots, |I|\}$ to I such that $\bar{X}_{\pi_1(i)}(N_{\pi_1(i)})$ is nondecreasing in i.

Initialize the upper confidence limit $\hat{U} \leftarrow -\infty$ and the largest standard error associated with any scenario $\bar{s} := \max_{i=1,2,\ldots,k} s_i$.

Compute the smallest sample size associated with any scenario that survived screening, $N_{\text{hi}} := \min\{N_{\pi_1(1)}, N_{\pi_1(2)}, \ldots, N_{\pi_1(|I|)}\}$, and $z_{\text{hi}} := t_{1-\alpha_{\text{hi}},N_{\text{hi}}-1}$.

The following loop computes the upper confidence limit as a maximum of upper bounds associated with different numbers ℓ of scenarios that could be used in estimating ES. For $\ell = \ell_{\min}, \ell_{\min}+1, \ldots, \lceil kp \rceil$,

(a) Compute $\Delta(\ell)$ as in Equation (3.8) and $B_S(\ell) := \bar{s}\Delta(\ell)$ to bound standard error in estimating ES using ℓ scenarios..

(b) Set

$$\hat{U} \leftarrow \max\left\{\hat{U}, \max_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) + z_{\mathrm{hi}} B_S(\ell)\right\}. \tag{3.10}$$

The confidence interval given in Equations (3.4) and (3.5) is $[\hat{L}, \hat{U}]$.

The maximum in Equation (3.8) is computed by using the Newton method to solve its KKT conditions. An algorithm for the optimizations in Equations (3.9) and (3.10) is given in Baysal and Staum (2008). The t-quantiles $z_{\mathrm{lo}}(\ell)$ and z_{hi} may be replaced by normal quantiles when the second-stage sample sizes are sufficiently large, as they typically are.

3.3. Experimental Results

We tested the simulation procedures by producing a 90% confidence interval (CI) for $\mathrm{ES}_{0.99}$ in the examples described in Chapter 1.1 and 1.2. The error $\alpha = 10\%$ was decomposed into $\alpha_o = 5\%$ for the outer level, $\alpha_s = 2\%$ for screening, and $\alpha_{\mathrm{lo}} = \alpha_{\mathrm{hi}} = 1.5\%$ for the inner-level lower and upper confidence limits. In each experiment, we chose our procedure's parameters k and n_0 according to a method described in Chapter 4. We compare our procedure with the `plain` procedure in Chapter 2. We ran the `plain` procedure with the same number k of scenarios as our procedure. To compare the procedures, we evaluate their confidence intervals' coverage probabilities and mean widths given the same fixed budget. We ran the experiments on a PC with a 2.4 GHz CPU and 4 GB memory under 64-bit Red Hat Linux. The code was written in C++ and compiled by `gcc` 3.4.6.

Similar to results reported in Lan et al. (2007a), we found that the plain procedure and our procedure both had coverage probabilities greater than the nominal confidence

level of 90% as long as $k \geq 40/p$, where p is the tail probability under consideration. In these examples, $p = 1 - 0.99 = 0.01$.

The following figures report average CI widths for 20 independent runs of the procedures. The error bars in the figures provide 95% confidence intervals for the mean width of our procedure's CI. (The width of the CI produced by the plain procedure is less variable, so the error bars for the plain procedure were too small to display.) In each figure, a horizontal line represents 10% relative error, that is, its value is one tenth of $\mathrm{ES}_{0.99}$. We include the line for the purpose of comparing the CI widths to a rough measure of desirable precision. It would not be very useful to attain a relative error far less than 10% because of model risk: that is, risk management models are not generally accurate enough that precision better than, say, 1% would convey meaningful information. On the other hand, if the CI width is much greater than 10% relative error, then the simulation experiment has left us with a great deal of uncertainty about the magnitude of $\mathrm{ES}_{0.99}$. For these reasons, we ran experiments with computational budgets such that our procedure yields CI widths in the neighborhood of 10% relative error.

Figure 3.2 shows how average CI width varies with a computational budget of C replications for the example of selling a put option described in Section 1.1. The much narrower CI widths achieved by our procedure show that the benefit of screening in directing more replications to important scenarios outweighs the cost of restarting and throwing out first-stage replications. In these experiments, our procedure produced a CI up to 116 times narrower than that produced by the plain procedure. On the log-log plot in Figure 3.2, the CI width decreases roughly linearly in the budget, with slope about -0.4 or -0.44. This is unfavorable compared to the usual $\mathcal{O}(C^{-1/2})$ order of convergence

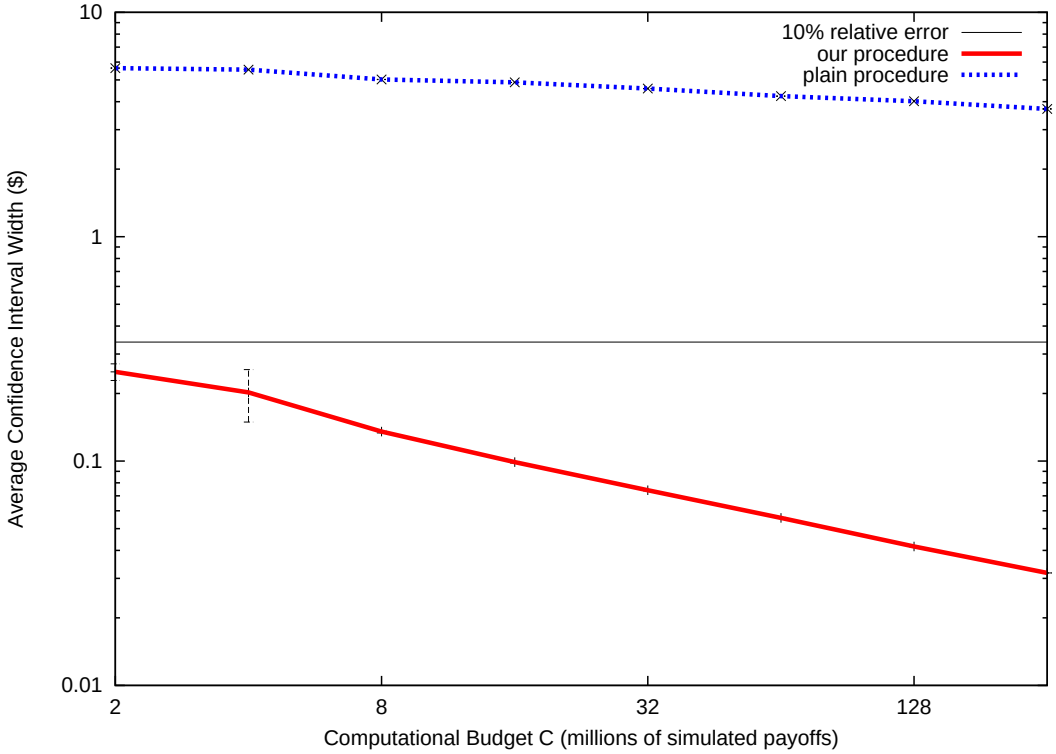

Figure 3.2. Average confidence interval width in the example of Section 1.1 given a fixed budget of simulation replications.

of ordinary Monte Carlo, but favorable compared to the $\mathcal{O}(C^{-1/3})$ order of convergence for a two-level simulation estimator of VaR found by Lee (1998) or the $\mathcal{O}(C^{-1/4})$ order of convergence for the procedure we proposed in Chapter 2.

Figure 3.3 shows similar results from the example of an options portfolio described in Section 1.2. In this example, larger computational budgets are required to get an precise estimate of ES. Again, our procedure produced CIs narrower than those from the plain procedure, up to a factor of 14. For low budgets, our procedure's advantage was not as great. For example, when C is 32 million, our best choice was $k = 4000$ and $n_0 = 4703$, so that more than half the budget was used up in the first stage before restarting, yet

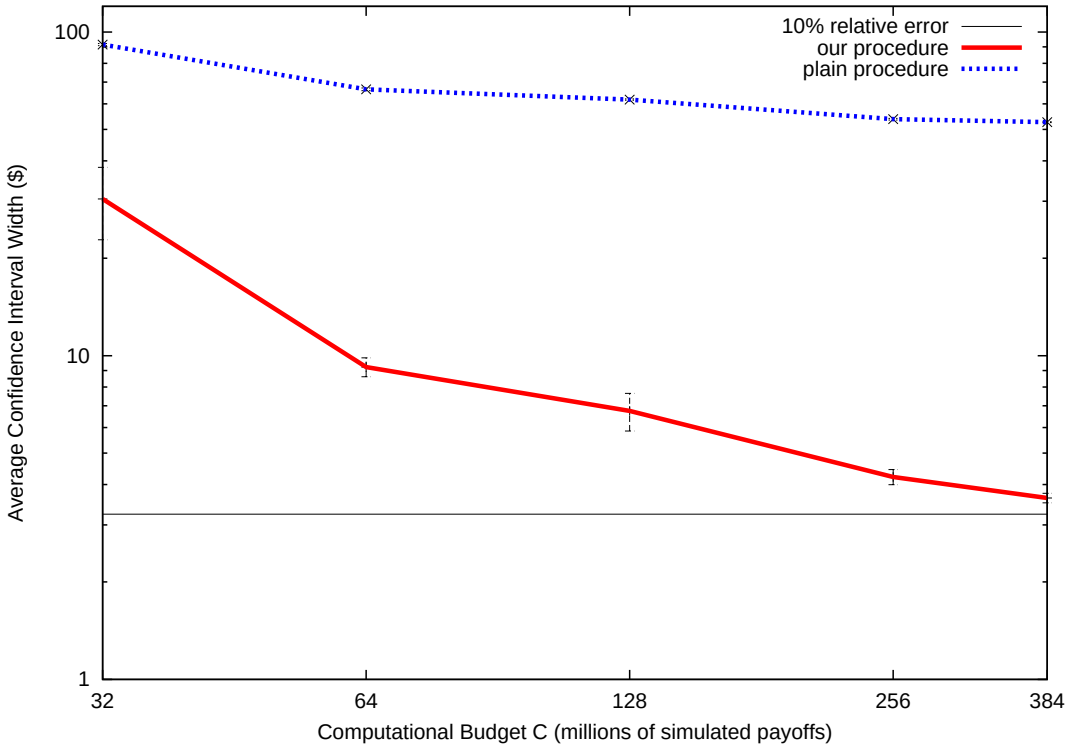

Figure 3.3. Average confidence interval width in the example of Section 1.2 given a fixed budget of simulation replications.

the first stage was too small to enable the procedure to screen out most of the scenarios that do not belong to the tail. When the computational budget is small, our procedure may not be able to produce a CI narrow enough to be useful. A multi-stage screening procedure (similar to Lesnevski et al., 2007, 2008) might overcome this problem.

Next we present results when the computational budget limits computing time. The budget constraint is implemented not by dynamically terminating the procedures when a given amount of clock time has elapsed, but by choosing values of k and n_0 such that the procedure takes approximately the given amount of time. Our procedure's running time is slightly variable, but all experiments' durations were within 5% of the allotted

time. A budget expressed in computing time is less favorable to our procedure (relative to the plain procedure) than a budget for the total number of replications: our procedure can spend a substantial amount of time in performing comparisons between scenarios as part of screening, even though it does not generate more replications then. The amount of time spent on screening when there are k scenarios is $\mathcal{O}(k^2)$ because there are $k^2/2$ pairs of scenarios that can be compared. This pushes us to choose smaller values of k (see Chapter 4). For instance, in the example of a single put option (Section 1.1), our procedure attains a CI width around \$0.0427 with a budget of $C = 120$ million replications or $T = 1,560$ seconds, but if the budget is in replications then we choose k to be about 600,000 scenarios, whereas if the budget is in computing time, we choose k to be about 427,000 scenarios. For budgets so large as to lead to choosing a very large k, the advantage of our procedure degrades. This can be seen in Figure 3.4, where the curve representing our procedure's CI width becomes flatter as the computing time T grows. Still, Figures 3.4 and 3.5 show that our procedure performs much better than the plain procedure when they are given equal computing times, producing a CI narrower by a factor of as much as 15 or 12 in these two examples.

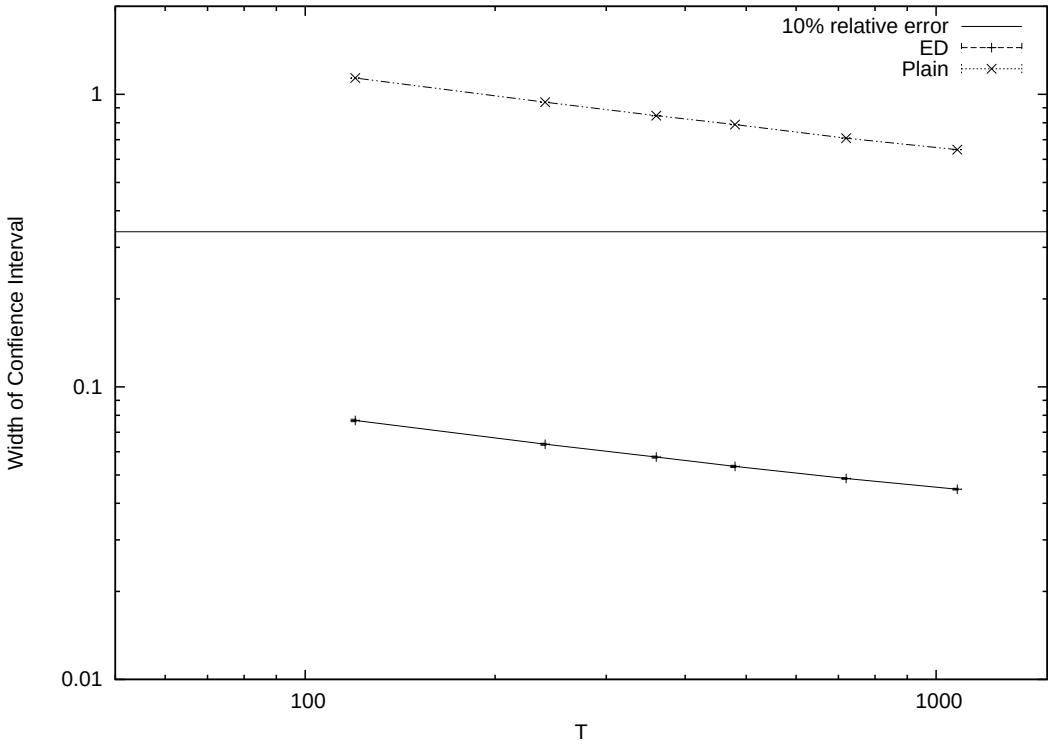

Figure 3.4. Average confidence interval width in the example of Section 1.1 given a fixed budget of computing time.

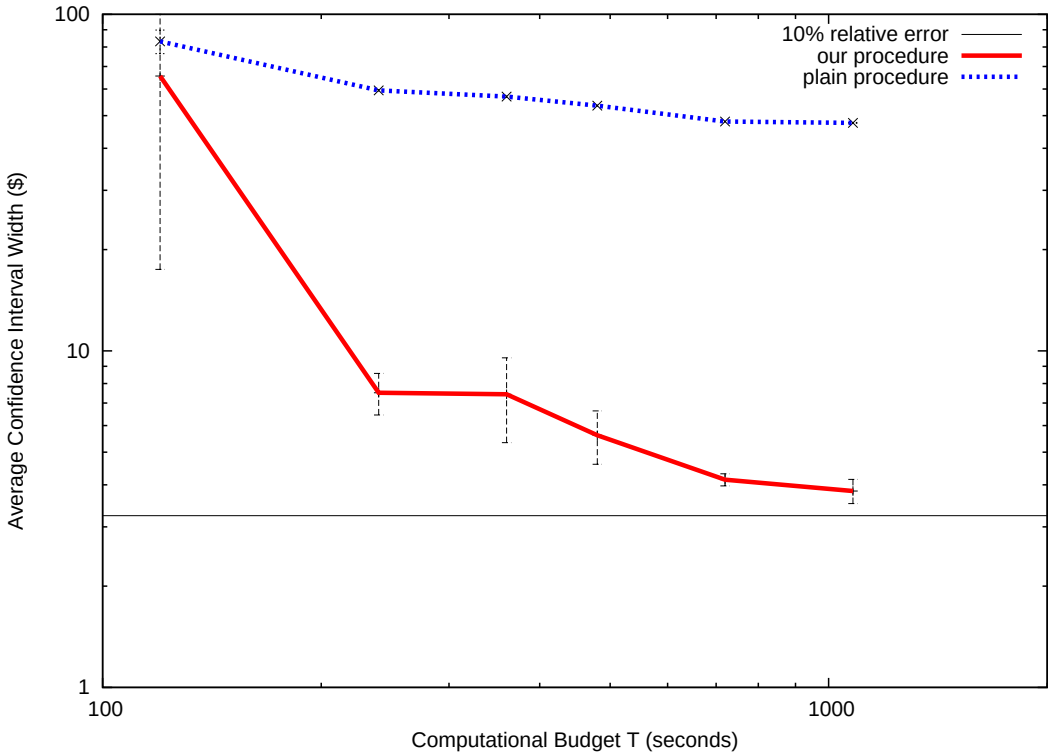

Figure 3.5. Average confidence interval width in the example of Section 1.2 given a fixed budget of computing time.

CHAPTER 4

Balancing Inner and Outer Level Simulation

Tuning the parameters of the two-level simulation with screening is important to its efficiency. Two of the parameters, the number of scenarios generated in the outer level k and the sample size for screening n_0, are the most important parameters for the performance of the two-level simulation. The number of scenarios k represents the effort spent on simulating the market movement at a future time T. The uncertainty about the future of the market is the source of the risk. Usually we want k to be large.

The sample size for screening (also called the first-stage sample size), n_0, is critical to the performance of the screening algorithm, and is strongly related to the second stage sample size $N_i,\ i \in I$ when the total simulation budget (the total number of simulated payoffs C or the total simulation time T), is fixed. The second-stage sample size N_i is the effort we spend on estimating the value of the portfolio conditioned on the scenarios that are important to the computation of loss. Again, $N_i, \forall i \in I$ are desired to be large, but they are not directly controllable; instead, they are determined indirectly by the number of scenarios k and the first-stage sample size n_0. Roughly speaking, N_i's are decreasing in k; but they are not monotone in n_0. In all, the two parameters (k, n_0) determine the effort spent on exploring market movement and on estimating the portfolio's values conditioned on the generated scenarios. Our goal is to improve the performance of the two-level simulation by tuning (k, n_0) such that exploration of new scenarios and estimating the portfolio's values conditioned on the generated scenarios are well balanced.

In the research field of simulation, there is a long history of research on the "bias-variance trade-off." If our procedure did not contain a screening algorithm, then there would be no n_0 to be set and choosing the number of scenarios k is a typical problem of "bias-variance trade-off." But the screening algorithm improves the efficiency of the two-level simulation in Section 3.3. If, on the other hand, the number of scenarios k were fixed, then screening is a traditional topic in the simulation literature of ranking and selection. Here we take k sampled scenarios as the alternatives and the $\lceil kp \rceil$ scenarios with the least $\lceil kp \rceil$ conditional portfolio values as the "best" configurations to be selected. In this literature, setting the sample size for screening has been studied in several different aspects. Stein (1945) proposed to determine the sample size for fixed width confidence interval estimation through a two-stage procedure. Rinott (1978) computed the sample size of the second stage of a two-stage procedure to guarantee the probability of correct selection of the best. When a fixed total number of allowable samples and all alternatives are given, Chen et al. (2000) proposed to compute the sample size of each alternative that maximizes the probability of correct selection of the best.

In our research, the number of scenarios, k, and the sample size, n_0, are chosen together to minimize the forecasted confidence interval width of expected shortfall through the two-level simulation. The sample size n_0, which is universal to all alternatives, is selected to help screen out the scenarios not in the tail, while at the same time it is important to keep n_0 as small as possible to save computational budget for valuing the retained scenarios.

Denote the expected width of the confidence interval generated through the two-level simulation as W, or $W(k, n_0)$. The latter notation explicitly shows the relationship between the parameters, k and n_0, and the expected width of the confidence interval. To

find the optimal parameters k and n_0 given the computational budget (simulation time T or total number of payoffs allowed to be simulated C), we first consider the following stochastic nonlinear integer problem.

$$\begin{aligned} \min \quad & W(k, n_0) \\ \text{s.t.} \quad & \Gamma\left(kn_0 + \sum_{i=1}^{|I|} N_i, C\right) \leq 0 \text{ or } \Gamma(T_r(k, n_0), T) \leq 0 \\ & n_0 \geq n_{\min}, k \geq k_{\min} \\ & k, n_0 \text{ positive integers} \end{aligned}$$

where $\Gamma(a, b)$ is a function used to express the constraint on the random computational budget the simulation may cost given (k, n_0), $k_{\min}, n_{\min}$ are the minimum values sufficient for the validity of the two-level simulation, $|I|$ as defined before, is the number of scenarios after screening, and $T_r(k, n_0)$ is the simulation time given (k, n_0). When $\Gamma(a, b) = \mathbf{E}(a) - b$, the expected computational budget is set to be less than the given budget; and when $\Gamma(a, b) = \mathbf{P}(a > b) - 0.01$, the probability that a simulation is accomplished within given computational budget is required to no less than 0.99. According to Section 3.3, $k_{\min} = \lceil 40/p \rceil$ and $n_{\min} = 30$. It is not possible to solve the problem because the expectations cannot be evaluated. Solving it empirically by actually performing two-level simulations at many values of (k, n_0) is not practical as these runs are very time consuming.

We need a simple and workable approach to address the challenge. Our strategy is to run a quick trial of two-level simulation and predict $W(k, n_0)$ based on the information collected from the trial. We denote the simulated data from the trial with k_0 scenarios and

n_{00} first-stage samples as $\mathcal{T}(k_0, n_{00})$. The two-level simulation procedure is complicated: empirical likelihood estimation and normal-theory inference are applied to deal with the uncertainties from the outer level and inner level respectively; efficiency techniques, such as screening and CRN, are introduced to improve the simulation efficiency. All these make the prediction of the confidence interval width difficult. Instead of a rigorously justified predictor, we provide two heuristic predictors based on necessary simplifications and approximations, which work well for all the experiments we tested.

The two heuristic predictors are adopted to forecast the output confidence interval width and the computational budget that would be consumed with given (k, n_0). One is an approximate most-likely predictor, $\mathcal{P}_1$; the other is a pessimistic predictor, $\mathcal{P}_2$, used to predict the "unluckily-wide" output and the "unluckily-expensive" budget that would be consumed in the simulation. Our decision rule mainly consists of the following two steps. First we solve a nonlinear programing problem using the approximate most-likely predictor $\mathcal{P}_1$ to find an optimal solution (k^*, n_0^*). Then within a certain neighborhood of the optimal solution (k^*, n_0^*), we try to solve another nonlinear programing problem with the pessimistic predictor $\mathcal{P}_2$ as the objective function and constraint function instead. That is, we want to set the procedure parameters so that they are not only suitable for the most-likely case but also acceptable for some unfavorable cases.

4.1. Approximation of W

To demonstrate how to set up the predictors, we first decompose the expected width of output confidence interval W into three parts: a part due to the outer-level uncertainty, W_1, a part due to the inner level uncertainty, W_2 and the rest of it, W_3. By assuming the

total number of simulated payoffs in the second stage, C_1, and the number of scenarios retained after screening, $K_1(k, n_0)$, are known, we approximate W_1 and W_2 by $\hat{W}_1$ and $\hat{W}_2$ respectively in Section 4.1.1 and 4.1.2.

The remaining part W_3 results from the discrepancy between two orderings of scenarios: the ordering of the scenarios right after screening, and the ordering of scenarios after second-stage sampling. It seems difficult to find a very accurate approximation of W_3. Yet from empirical study, compared with W_1 and W_2, the remaining part W_3 is typically negligible. The reason is that when n_0 becomes large enough,the first stage makes the ordering of the retained scenarios close to the true ordering. Meanwhile, when enough computational budget is reserved for the second-stage sampling,the ordering of the scenarios after the second sampling should be very close to the true ordering too. In such case W_3 becomes negligible. On the other hand, if k or n_0 is set to some unsuitable values, i.e. k is extremely large or n_0 is relatively small or unnecessarily large, W_3 becomes relatively large and even dominates $W_1 + W_2$. This indicates that W_3 works like a penalty function, forcing k and n_0 to take values within some region close to their optimum. Due to this fact, W_3 can be substituted by a penalty function $\hat{W}_3$, which vanishes when the two aforementioned orderings agree and is large when the two orderings are dissimilar.

The two predictors $\mathcal{P}_1$ and $\mathcal{P}_2$ are designed to predict $K_1(k, n_0)$ and C_1 in most likely and pessimistic cases. The ratio $K_1(k, n_0)/k$ stands for the effectiveness of the screening algorithm and the total number of simulated payoffs in the second stage C_1 indicates the time efficiency of the screening algorithm when total allowable running time T is fixed. The estimation of $K_1(k, n_0)$, C_1, and the predictors $\mathcal{P}_1$ and $\mathcal{P}_2$, will be addressed in Section 4.2.

When we adopt the predictor $\mathcal{P}_1$,there is an important assumption upon which we heavily rely.

Assumption 1. *For a relatively wide region of the parameter pairs of* (k, n_0) *around the optimal settings with certain given computational budget, the performance of the simulation remains stable; that is when computation budget,* C *or* T*, and parameters* (k, n_0) *are given, for different runs of the simulation, the output confidence interval width and the internal variables* $K_1(k, n_0)$ *and* C_1 *have low variances such that they can be treated as constants.*

For the simulation we are studying, the robustness of simulated output to the procedure parameters is often observed. That makes the Assumption 1 reasonable which facilitates our analysis.

Let us decompose $W = \mathbf{E}\left\{\hat{U} - \hat{L}\right\}$ into $W = W_1 + W_2 + W_3$, as described above, in the following forms.

$$
\begin{aligned}
W &= \mathbf{E}\left\{\left[\sum_{i=1}^{\bar{\bar{l}}} \bar{\bar{w}}_i' \bar{X}_{\pi_{1i}}(N_i) + z_{\mathrm{hi}} B_S(\bar{\bar{l}})\right] - \left[\sum_{i=1}^{\tilde{\underline{l}}} \underline{\tilde{w}}_i' \bar{X}_{\pi_{0i}}(N_i) + z_{\mathrm{lo}}(\tilde{\underline{l}}) B_0(\tilde{\underline{l}})\right]\right\} \\
&= \underbrace{\mathbf{E}\left\{\sum_{i=1}^{\bar{\bar{l}}} \bar{\bar{w}}_i' \bar{X}_{\pi_{1i}}(N_i) - \sum_{i=1}^{\check{\underline{l}}} \underline{\check{w}}_i' \bar{X}_{\pi_{1i}}(N_i)\right\}}_{W_1} \qquad (4.1) \\
&\quad + \underbrace{\mathbf{E}\left\{z_{\mathrm{hi}} B_S(\bar{\bar{l}}) + z_{\mathrm{lo}}(\tilde{\underline{l}}) B_0(\tilde{\underline{l}})\right\}}_{W_2} \qquad (4.2) \\
&\quad + \underbrace{\mathbf{E}\left\{\sum_{i=1}^{\check{\underline{l}}} \underline{\check{w}}_i' \bar{X}_{\pi_{1i}}(N_i) - \sum_{i=1}^{\tilde{\underline{l}}} \underline{\tilde{w}}_i' \bar{X}_{\pi_{0i}}(N_i)\right\}}_{W_3} \qquad (4.3)
\end{aligned}
$$

where

$$\underline{\tilde{w}} := \arg\min_{w\in\mathcal{S}} \sum_{i=1}^{k} w_i' X_{\pi_{0i}}(N_i) \qquad \underline{\tilde{l}} := \arg\min_{l=\lfloor kp \rfloor}^{\ell_{\max}} \min_{w\in\mathcal{S}(l)} \sum_{i=1}^{l} w_i' X_{\pi_{0i}}(N_i)$$

$$\underline{\check{w}} := \arg\min_{w\in\mathcal{S}} \sum_{i=1}^{k} w_i' X_{\pi_{1i}}(N_i) \qquad \underline{\check{l}} := \arg\min_{l=\lfloor kp \rfloor}^{\ell_{\max}} \min_{w\in\mathcal{S}(l)} \sum_{i=1}^{l} w_i' X_{\pi_{1i}}(N_i)$$

$$\bar{\check{w}} := \arg\max_{w\in\mathcal{S}} \sum_{i=1}^{k} w_i' X_{\pi_{1i}}(N_i) \quad \bar{\check{l}} := \arg\max_{l=\ell_{\min}}^{\lceil kp \rceil} \max_{w\in\mathcal{S}(l)} \sum_{i=1}^{l} w_i' X_{\pi_{1i}}(N_i).$$

Despite the messy expressions, the underlying reason for this decomposition is pretty clear. The first part W_1 is constructed as the expected width of a confidence interval from empirical likelihood estimation on the data set $X_{\pi_{1i}}(N_i), i = 1, 2, \ldots, |I|$. The second part W_2 involves bounds on standard deviations associated with inner-level uncertainty. In any case, the number of scenarios retained after screening $|I|$ is larger than $\ell_{\max}$. Therefore, when empirical likelihood estimation is applied to construct the confidence interval of ES, selection bias is inevitable. To cancel it, another permutation of $X_i(N_i), i \in I$, $\pi_{0i}, i = 1, 2, \ldots, |I|$, is adopted to compute the lower bound of ES instead of the natural one, π_{1i}. The last part W_3 is designed to capture the effect of replacing π_{1i} with π_{0i} in the computation of the lower confidence bound of ES.

Based on the following analysis, we can approximate W by

$$\begin{aligned}\hat{W}(k,n_0) \;:=\; & \frac{E_o}{\sqrt{k}} + \\ & \sqrt{\frac{K_1(k,n_0)}{C_1}}\left(\mathrm{E}\left[\sqrt{\sigma^2+\nu^2 Z^{(1)}(K_1(k,n_0))}\right] z_{1-\alpha_{\mathrm{lo}}}\Delta(\lceil kp\rceil)\right) \\ & +\sqrt{\frac{K_1(k,n_0)}{C_1}}\left(\mathrm{E}\left[\sqrt{\sigma^2+\nu^2 Z^{(1)}(\lceil kp\rceil)}\right] z_{1-\alpha_{\mathrm{hi}}}\Delta(\ell_{\min})\right) \\ & +\frac{1}{\underline{l}}\sum_{(i,j)\in J}\left\{\frac{\sigma^*_{ij}}{\sqrt{2\pi}}\exp\left\{-\frac{d_{ij}^2}{2\sigma^{*2}_{ij}}\right\} + d_{ij}\left(\Phi(-\frac{d_{ij}}{\sigma'_{ij}}) - \Phi(-\frac{d_{ij}}{\sigma^*_{ij}})\right)\right\}. \end{aligned} \tag{4.4}$$

The next three sub-sections define the notation and provide intuition for the equation above.

4.1.1. Approximation of W_1

The width of the outer-level empirical likelihood confidence interval W_1 is an unknown function of k. From empirical observations, we can assume that

Assumption 2. *Empirical likelihood confidence interval widths are* $\mathcal{O}(k^{-1/2})$.

Let $E_o/\sqrt{k_0}$ be the width of the outer-level empirical likelihood confidence interval in $\mathcal{T}(k_0, n_{00})$. Based on Assumption 2, the width when there are k scenarios will be approximately $E_o/\sqrt{k}$. We estimate E_o by

$$\sqrt{k_0}\left(\max_{\mathbf{w}\in\mathcal{S}(k_0)}\sum_{i=1}^{k_0} w'_i\bar{X}_{(i)}(n_{00}) - \min_{\mathbf{w}\in\mathcal{S}(k_0)}\sum_{i=1}^{k_0} w'_i\bar{X}_{(i)}(n_{00})\right)$$

where $\bar{X}_{(i)}(n_{00})$ is the ith order statistic of the sample averages in $\mathcal{T}(k_0, n_{00})$.

4.1.2. Approximation of W_2

First we make some simplifications regarding the second-stage sample errors. Let $s_i^2 := S_i^2(N_i)/N_i$. As N_i is chosen to be proportional to the first-stage sample variance, $s_i^2 C_1/K_1(k,n_0) = (S_i^2(N_i)/S_i^2(n_0))(\sum_{j\in I} S_j^2(n_0)/K_1(k,n_0))$. By Assumption 1, $\sum_{j\in I} S_j^2(n_0)/K_1(k,n_0)$ is the average of the first-stage sample variances $S_j^2(n_0), j \in I$. It is approximated as $\sum_{i=1}^{K_0} S_{(i)}^2(n_{00})/K_0$, where K_0 is the nearest integer to $K_1(k,n_0)k_0/k$. Conditioning on N_i, $S_i^2(N_i)/S_i^2(n_0)$ obeys F_{N_i-1,n_0-1} distribution. To speed up computation, s_i^2 is simplified as a two-moment normal approximation, as shown in Assumption 3.

Assumption 3. *s_i^2 is normally distributed with mean σ^2 and variance ν^2, where*

$$\begin{aligned}
\sigma^2 &= \frac{(n_0-1)\sum_{i=1}^{K_0} S_{(i)}^2(n_{00})}{(n_0-3)K_0}, \\
\nu^2 &= \frac{2(\sigma^2)^2(\bar{N}+n_0-4)}{(\bar{N}-1)(n_0-5)}, \\
\bar{N} &= \frac{C_1}{K_1(k,n_0)}.
\end{aligned}$$

Now let us deal with the other two factors in W_2, $B_S(\bar{\bar{l}})$ and $B_0(\tilde{\underline{l}})$. The undetermined variables $\bar{\bar{l}}$ and $\tilde{\underline{l}}$, standing for the number of scenarios used to construct the upper and lower bound estimation, are assumed to be constants as a consequence of Assumption 1. Also because $B_S(\bar{\bar{l}})$ and $B_0(\tilde{\underline{l}})$ are insensitive to the values of $\bar{\bar{l}}$ and $\tilde{\underline{l}}$, the following assumption is natural.

Assumption 4. *The number of scenarios used to construct the upper bound in empirical likelihood estimation, $\bar{\bar{l}}$, and the lower bound, $\tilde{\underline{l}}$, equal to ℓ_{min} and $\lceil kp \rceil$, respectively.*

Based on assumptions above, with additivity and scalability of maximum operation, W_2 can be approximated by

$$\begin{aligned} &\sqrt{\frac{K_1(k,n_0)}{C_1}}\left(\mathrm{E}\left[\sqrt{\sigma^2+\nu^2 Z^{(1)}(K_1(k,n_0))}\right] z_{1-\alpha_{\mathrm{lo}}}\Delta(\lceil kp\rceil)\right) \\ &+\sqrt{\frac{K_1(k,n_0)}{C_1}}\left(\mathrm{E}\left[\sqrt{\sigma^2+\nu^2 Z^{(1)}(\lceil kp\rceil)}\right] z_{1-\alpha_{\mathrm{hi}}}\Delta(\ell_{\min})\right) \end{aligned} \tag{4.5}$$

where $Z^{(1)}(n)$ is the maximum of n i.i.d standard normal random variables. The expectation can be computed by numerical integration and $\Delta(\lceil kp\rceil)$ and $\Delta(\ell_{\min})$ can be computed through the approach described in Appendix A.

4.1.3. Substitute for W_3

As mentioned above, for W_3, the remaining part of W, instead of looking for an approximation, we find a substitute for it. The substitute, denoted as $\hat{W}_3$, changes in a way similar to W_3 with respect to k and n_0. When k and n_0 are close to the optimal setting so that the screening algorithm can efficiently eliminate most inferior scenarios and the order of scenarios in set I right after screening and that after the second-stage sampling are similar, $\hat{W}_3$ is almost negligible as W_3 is. When k or n_0 takes unsuitable values, the substitute $\hat{W}_3$, although it may be quite different from W_3, still dominates $W_1 + W_2$.

The intuition for $\hat{W}_3$ comes from a simple case with two scenarios. In such a case, let μ_1, μ_2 be the conditional portfolio value for scenarios 1 and 2. After the first-stage sampling, the sample averages $\bar{X}_1(n_0)$ and $\bar{X}_2(n_0)$ approximately obey $N(\mu_1, \sigma_1^2/n_0)$ and $N(\mu_2, \sigma_2^2/n_0)$, respectively. The variances σ_1^2 and σ_2^2 are determined by the randomness in the simulation conditioned on the corresponding scenarios. The covariance σ_{12}^2 is also

needed due to CRN at the first stage. All these parameters can be estimated from the pilot information $\mathcal{T}(k_0, n_{00})$. Accordingly, after the second stage, the sample averages are asymptotically distributed as $N(\mu_i, \sigma_i^2/N_i), i = 1, 2$. For notational simplicity, denote $d_{12} = \mu_1 - \mu_2$, $\sigma'_{12} = \sigma_{12}\sqrt{1/n_0}$, $\sigma^*_{12} = \sqrt{\sigma_1^2/N_1 + \sigma_2^2/N_2}$, $D_1 = \bar{X}_1(n_0) - \bar{X}_1(n_0)$ and $D_2 = \bar{X}_1(N_1) - \bar{X}_2(N_2)$. Conditioning on n_0 and $N_i, i = 1, 2$, $D_1 \sim N(d_{12}, \sigma'^2_{12})$ and $D_2 \sim N(d_{12}, \sigma^{*2}_{12})$. Ignoring the uncertainty on $N_i, i = 1, 2$, we can approximate $\bar{X}_{\pi_1(1)}(N_{\pi_1(1)}) - \bar{X}_{\pi_0(1)}(N_{\pi_0(1)})$ by

$$\begin{aligned} & \mathrm{E}[|D_2| \cdot \mathbf{1}\{D_1 D_2 < 0\}] \\ = \; & \mathrm{E}[D_2 \mathbf{1}\{D_2 > 0\}\mathbf{1}\{D_1 < 0\}] \\ & + \mathrm{E}[-D_2 \mathbf{1}\{D_2 < 0\}\mathbf{1}\{D_1 > 0\}] \\ = \; & \frac{\sigma^*_{12}}{\sqrt{2\pi}} \exp\left\{-\frac{d_{12}^2}{2\sigma^{*2}_{12}}\right\} + d_{12}\left(\Phi(-\frac{d_{12}}{\sigma'_{12}}) - \Phi(-\frac{d_{12}}{\sigma^*_{12}})\right), \end{aligned}$$

where $\Phi(\cdot)$ is the c.d.f of the standard normal distribution. The equality at the end is a consequence of the fact that D_1 and D_2 are independent.

Then, the substitute $\hat{W}_3$ can be written as

$$\hat{W}_3(k, n_0) := \frac{1}{\underline{l}} \sum_{(i,j)\in J} \left\{ \frac{\sigma^*_{ij}}{\sqrt{2\pi}} \exp\{-\frac{d_{ij}^2}{2\sigma^{*2}_{ij}}\} + d_{ij}\left(\Phi(-\frac{d_{ij}}{\sigma'_{ij}}) - \Phi(-\frac{d_{ij}}{\sigma^*_{ij}})\right)\right\}$$

where $\underline{l} = l^* k/k_0$ and $J = \{(\underline{l} - i, \underline{l} + i + 1) | i = 0, \cdots, \underline{l} \wedge (K_1(k, n_0) - \underline{l}) - 1\}$. The parameter l^* is the number of sampled conditional portfolio values selected to construct the lower bound of ES in the pilot simulation. The substitute $\hat{W}_3$ is set up based on the simplification that $\check{\underline{l}} = \tilde{\underline{l}} = \underline{l}$, $\check{\underline{w}} = \tilde{\underline{w}} = \{1/\underline{l}, \cdots, 1/\underline{l}\}$ and any pair of scenarios within J behaves exclusively and independently in contributing to W_3. Again, the substitute $\hat{W}_3$

is not an approximation of W_3. It just plays a similar role in $\hat{W}$ as W_3 does in W with respect to the two procedure parameters k and n_0.

4.2. Measurement and Prediction

In Equation (4.4), most terms can be directly estimated or approximated from the pilot information, $\mathcal{T}(k_0, n_{00})$, but the number of scenarios retained after screening $K_1(k, n_0)$ and the number of allocated simulated payoffs at the second stage C_1 need specific consideration. When the computational budget is given in total simulated payoffs C, C_1 can be easily calculated by $C_1 = C - kn_0$. When the computational budget is given by total allowable running time T, we have to do more work to estimate the running time of the two-level simulation with parameters k and n_0.

4.2.1. When Computational Effort is Given by T

In this case, the number of allocated simulated payoffs for the second stage C_1 has to be calculated by estimating the time left for the second stage sampling T_1 and the unit time required for the sampling of each payoff R_{sam}. First we need to use a high precision timer provided by the operating system and assume that the running environment of our procedure is unchanged between the pilot simulation and the experiment we would like to run. Then we need to analyze the computational complexity of the two-level simulation.

In Figure 4.1, the first row of the diagram shows the main time-expensive steps of the two-level simulation. The second row shows the corresponding approximation formula for time spent on each step. All the constant multipliers (the R's) are determined from the observed computation time in the historical or pilot simulation. Some of these are derived

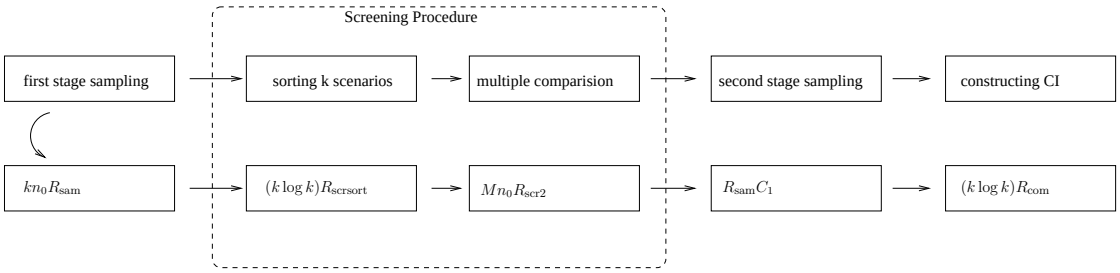

Figure 4.1. Time-consuming Steps of Two-Level Simulation

from the expected computational complexity. For example, the time spent on "sorting" has the expected computational complexity of $\mathcal{O}(k\log k)$ as Quick Sort is adopted, and we may then approximate the time spent on "sorting" as $(k\log k)R_{\text{scrsort}}$, where R_{scrsort} is a constant multiplier. Similar reasoning accounts for the first-stage sampling and second-stage sampling approximation formulas, where R_{sam} is the average time spent on the simulation of unit payoff. The time spent on constructing the confidence interval is $(k\sqrt{k})R_{\text{com}}$. A rough intuition is that the computation of the lower or upper bound limit of $\sum_{w\in\mathcal{S}(l)} w'\bar{X}_{\pi.i}(N_i)$ is $\mathcal{O}(k)$ for each $l\in[\ell_{\min},\ell_{\max}]$, and $\ell_{\max}-\ell_{\min}$ is asymptotically $\mathcal{O}(\sqrt{k})$. The time spent on the multiple comparisons of the screening procedure is a function of k and n_0, and is monotone in k. But the relationship between the time spent on the multiple comparisons and the first-stage sample size n_0 is complex, due to the fact that increasing n_0 can make it easier to screen out scenarios. What this means is that, the number of comparisons need to be done, denoted as M, decreases or remains unchanged as n_0 increases; meanwhile, for each comparison, the computational effort on computing the variance $S_{ij}^2(n_0)$, which is dominant in the total computational effort of each comparison, is a linearly increasing function of n_0. We leave the estimation of M to Section 4.2.2. If the number of comparisons M is already known, the time spent on multiple comparisons

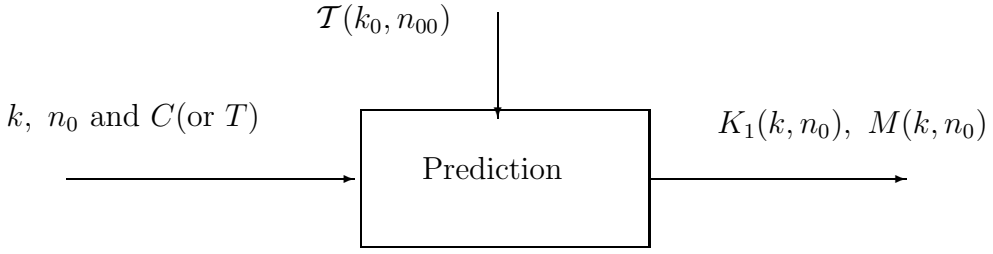

Figure 4.2. Function of the Predictor of Screening

of the screening procedure can be approximated as Mn_0R_{scr2}, where n_0R_{scr2} stands for average time on the calculation of each $S^2_{ij}(n_0)$, and R_{scr2} is a constant multiplier.

Based on the approximations above, we can estimate C_1 as follows:

$$(4.6)\quad C_1 = \frac{T_1}{R_{\text{sam}}} = \frac{T - kn_0R_{\text{sam}} - kR_{\text{scr}} - (k\log k)R_{\text{scrsort}} - (k\sqrt{k})R_{\text{com}} - Mn_0R_{\text{scr2}}}{R_{\text{sam}}},$$

where the number of comparisons M is still unknown. The number of scenarios retained after screening, $K_1(k, n_0)$, is also an unknown variable that needs to be predicted.

4.2.2. Predictors

Figure 4.2.2 shows the problem we discuss in this section: predicting the number of scenarios after screening $K_1(k, n_0)$ and the number of comparisons to be done $M(k, n_0)$ when the computational effort is given by simulated payoffs C (or clock time T). A predictor of the screening procedure is proposed as a function that takes $\mathcal{T}(k_0, n_{00})$ and any pair of (k, n_0) as inputs. The outputs are $K_1(k, n_0)$ and C_1. For simplicity and computational efficiency, based on the pilot information $\mathcal{T}(k_0, n_{00})$ for any pair of parameters (k, n_0), we suppose all the k scenarios are re-sampled from the k_0 scenarios contained in $\mathcal{T}(k_0, n_{00})$ equally; i.e., for each scenario there are k/k_0 duplications in the prediction. Similarly,

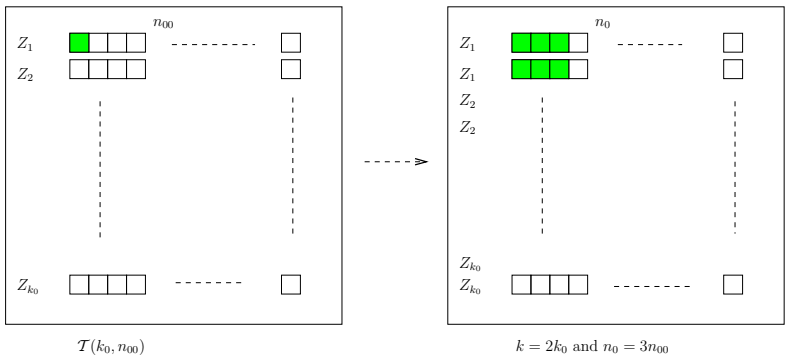

Figure 4.3. Re-sampling Based on $\mathcal{T}(k_0, n_{00})$ for $k = 2k_0$ and $n_0 = 3n_{00}$

all payoffs conditioned on each scenario in the prediction are re-sampled from the pilot information $\mathcal{T}(k_0, n_{00})$ equally again. Each $X_{ij} \in \mathcal{T}(k_0, n_{00}),\ i = 1, \cdots, k_0\ j = 1, \cdots, n_{00}$ is assumed to repeat $(k/k_0)(n_0/n_{00})$ times in the prediction. Figure 4.3 shows that when $k = 2k_0$ and $n_0 = 3n_{00}$, according to our re-sampling scheme, each payoff in $\mathcal{T}(k_0, n_{00})$ repeats 6 times, as the colored boxes indicate. It is obvious that the above resampling does not really needed to be done; indeed, we only need to predict $\bar{X}_i(n_0), S_i^2(n_0), i = 1 \cdots, k$, etc. on the basis of this re-sampling scheme. Then parameters in Equation (4.4), i.e. d_{ij}, σ_{ij}^* and $\sigma_{ij}^{'}$, can also be estimated based on these predictions.

4.2.2.1. The Most-likely Predictor $\mathcal{P}_1$. By applying the screening algorithm to the re-sampled data set, we can compute the number of comparisons M directly. As for the number of scenarios after screening $K_1(k, n_0)$, instead of assuming $X_i(n_0)$ constant, we take $X_i(n_0) \sim N(\mu_i, \sigma_i^2)$ and $A_{ji}(n_0) = \mathbf{1}\left\{\bar{X}_i(n_0) > \bar{X}_j(n_0) + d\sqrt{\frac{S_{ij}^2(n_0)}{n_0}}\right\}$ as a Bernoulli random variable with the probability of success $\Phi\left(d - \frac{\mu_i - \mu_j}{\sqrt{\frac{\sigma^{*2}_{ij}}{n_0}}}\right)$. The parameters μ_i, σ_i^2 and σ^{*2}_{ij} could be approximated from the pilot information $\mathcal{T}(k_0, n_{00})$ as mentioned above. By applying screening algorithm, scenario i is retained or not as determined by whether

$\sum_{j=1,j\neq i}^{k} A_{ji}(n_0) < \lceil kp \rceil$. From a normal approximation, we get

$$\mathrm{E}\left[\mathbf{1}\left\{\sum_{j=1,j\neq i}^{k} A_{ji}(n_0) < \lceil kp \rceil\right\}\right] \approx \Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right),$$

where

$$\begin{aligned}
\mu_B(i) &= \sum_{j=1,j\neq i}^{k_0} \frac{k}{k_0} \Pr\{A_{ji}(n_0)\} \\
\sigma_B^2(i) &= \sum_{j=1,j\neq i}^{k_0} \frac{k}{k_0} \Pr\{A_{ji}(n_0)\}(1 - \Pr\{A_{ji}(n_0)\}).
\end{aligned}$$

Thus, we have the approximation

$$K_1(k, n_0) \approx \sum_{i=1}^{k_0} \frac{k}{k_0} \Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right). \tag{4.7}$$

The following procedure describes how $K_1(k, n_0)$ and M are actually calculated.

(1) Feed $\bar{X}_i(n_{00}), S_i^2(n_{00}), \pi_0(i)$ $\forall i = 1, 2, \ldots, k_0$ and $S_{ij}^2(n_{00})$ for some i, j from $\mathcal{T}(k_0, n_{00})$[1]

(2) Take inputs k and n_0.

(3) Initialize $K_1 = \lceil kp \rceil$, $M' = 0$, $d = t_{n_0-1,1-\alpha_s/((k-\lceil kp \rceil)\lceil kp \rceil)}$, $\tilde{S}^2(n_{00}) = \max_{i=0}^{\lceil kp \rceil} S_{\pi_0(i)}^2(n_{00})$ and $i \leftarrow k_0$.

(a) *Pre-screening*: If $\bar{X}_{\pi_0(i)}(n_{00}) > \bar{X}_{\pi_0(\lceil kp \rceil)}(n_{00}) + d\sqrt{(S_{\pi_0(i)}^2(n_{00}) + \tilde{S}^2(n_{00}))/n_0}$, scenario $\pi_0(i)$ is pre-screened out: go to Step (3c).

(b) *Screening*: Initialize $c \leftarrow 0$, $b \leftarrow 0$, $\mu_B \leftarrow 0$, $\sigma_B^2 \leftarrow 0$, $e \leftarrow 0$ and $j \leftarrow 1$.

[1]Not all $S_{ij}^2(n_{00})$ are necessarily computed in the pilot simulation. For those with unknown value, the variance is computed inline when needed.

(i) If $S^2_{\pi_0(i)\pi_0(j)}$ is unassigned, compute the sample variance $S^2_{\pi_0(i)\pi_0(j)}$ of $X_{\pi_0(i)1} - X_{\pi_0(j)1}, X_{\pi_0(i)2} - X_{\pi_0(j)2}, \ldots, X_{\pi_0(i)n_0} - X_{\pi_0(j)n_0}$.

(ii) If $c < \lceil kp \rceil$, set $M' \leftarrow M' + 1$.

(iii) If $\bar{X}_{\pi_0(i)}(n_{00}) > \bar{X}_{\pi_0(j)}(n_{00}) + (d+2)S_{\pi_0(i)\pi_0(j)}(n_{00})/\sqrt{n_0}$, scenario $\pi_0(j)$[2] beats scenario $\pi_0(i)$ for sure: set $b \leftarrow b + \frac{k}{k_0}$.

Else if $\bar{X}_{\pi_0(i)}(n_{00}) > \bar{X}_{\pi_0(j)}(n_{00}) + (d-2)S_{\pi_0(i)\pi_0(j)}(n_{00})/\sqrt{n_0}$, scenario $\pi_0(j)$ beats scenario $\pi_0(i)$ with probability $a \leftarrow \Phi\left(d - \frac{\bar{X}_i(n_{00}) - \bar{X}_j(n_{00})}{\sqrt{\frac{S^2_{ij}(n_{00})}{n_0}}}\right)$: set $\mu_B \leftarrow \mu_B + \frac{k}{k_0}a$ and $\sigma_B^2 \leftarrow \sigma_B^2 + \frac{k}{k_0}a\,(1-a)$.

Otherwise, scenario $\pi_0(j)$ cannot beat scenario $\pi_0(i)$ for sure: set $e \leftarrow e + \frac{k}{k_0}$.

(iv) If $\bar{X}_{\pi_0(i)}(n_{00}) > \bar{X}_{\pi_0(j)}(n_{00}) + dS_{\pi_0(i)\pi_0(j)}(n_{00})/\sqrt{n_0}$, set $c \leftarrow c + \frac{k}{k_0}$.

(v) If $b \geq \lceil kp \rceil$, scenario $\pi_0(i)$ is screened out: go to Step (3c).

Otherwise, set $j \leftarrow j + 1$.

(vi) If $j < i$, go to Step (3(b)i).

Otherwise, go to Step (3c).

(c) If $b \geq \lceil kp \rceil$, scenario $\pi_0(i)$ is screened out for sure.

Else if $(j-1)\frac{k}{k_0} - e < \lceil kp \rceil$, scenario $\pi_0(i)$ can not be screened out for sure: set $K_1 \leftarrow K_1 + \frac{k}{k_0}$.

Otherwise, scenario $\pi_0(i)$ is retained with probability $\Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right)$: set $K_1 \leftarrow K_1 + \frac{k}{k_0}\Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right)$.

Loop: Set $i \leftarrow i - 1$. If $i > \lceil k_0 p \rceil$, go to Step (3a).

(4) set $K_1 \leftarrow K_1 \bigvee \ell_{\max}$.

[2]For time efficiency, we approximate $\Phi(x) = 1$ and $\Phi(-x) = 0$ when $x \geq 2$.

(5) set $M \leftarrow \frac{k^2}{k_0^2} M'$

For experiments in which the computational effort is measured by the total number of simulated payoffs, C, only $K_1(k, n_0)$ is significant. If computational time T is given, both $K_1(k, n_0)$ and M are important.

4.2.2.2. The Pessimistic Predictor $\mathcal{P}_2$. Due to the fact that optimizing n_0 tends to make it small (but still large enough to screen out a lot of inferior scenarios at the end of the first stage), the optimized parameters (k, n_0), say (k^*, n_0^*), are highly sensitive to the quality of the approximation, $\hat{W}(k, n_0)$. That is, n_0^* may be only slightly too small yet it may lead to dramatically bad performance for some runs of two-level simulations. A special step called "robustness" is added right after the optimized (k^*, n_0^*) is achieved to tailor (k^*, n_0^*) a little bit to stabilize the output performance of the simulation. To accomplish this goal, the pessimistic instead of most likely predictor is adopted. Unless explicitly pointed out, all assumptions we made in Section 4.2.2.1 are still applicable here.

From Section 4.2.2.1, the number of scenarios after screening $K_1(k, n_0)$ can be taken as the sum of Bernoulli random variables with probability of success equal to $\Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right)$. As shown in Equation (4.7), the expectation is taken as the most likely prediction of $K_1(k, n_0)$. As a pessimistic predictor, $\mathcal{P}_2$ takes the 95th percentile of $K_1(k, n_0)$ as the output. Similar to Equation (4.7), the 95th percentile of $K_1(k, n_0)$ can be approximated by the 95th percentile of the normal distribution with mean $\sum_{i=1}^{k_0} \frac{k}{k_0} \Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right)$ and variance $\sum_{i=1}^{k_0} \frac{k}{k_0} \Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right)\left(1 - \Phi\left(\frac{\lceil kp \rceil - 0.5 - \mu_B(i)}{\sigma_B(i)}\right)\right)$.

For the number of comparisons needed to be done, M, a similar method is applicable. Let $M = \sum_{i=\lceil kp \rceil + 1}^{k} M_{\pi_0(i)}$, where $M_{\pi_0(i)}$ is the number of comparisons needed to be done for the decision of scenario $\pi_0(i)$. As each comparison is taken as a Bernoulli random

variable, $M_{\pi_0(i)}$ is the smaller of $i-1$ or the number of comparisons needed to be done until the total number of successful comparisons is $\lceil kp \rceil$. For simplification, we assume all the Bernoulli random variables are independent with the probabilities of success jumping down from 1 to $\bar{p}_B$ and then finally to 0.[3] Obviously, $M_{\pi_0(i)}$ is now simplified as a linear transform of a truncated negative binomial random variable. Then the number of comparisons M can be estimated as

$$\mathrm{E}[M] = \sum_{i=\lceil kp \rceil + 1}^{k} \mathrm{E}[M_{\pi_0(i)}] \tag{4.8}$$

where

(4.9)

$$\mathrm{E}[M_{\pi_0(i)}] = \begin{cases} \lceil kp \rceil, & \text{when } b > \lceil kp \rceil; \\ i-1, & \text{when } i-e-1 < \lceil kp \rceil; \\ b + \dfrac{\lceil kp \rceil - b}{\bar{p}_B} \mathrm{NB}_{\bar{p}_B}(\lceil kp \rceil - b + 1, i - e - \lceil kp \rceil - 2) & \\ \quad + (1 - \mathrm{NB}_{\bar{p}_B}(\lceil kp \rceil - b, i - e - \lceil kp \rceil - 1))(i - b - 1) & \text{o.w.} \end{cases}$$

and $\mathrm{NB}_{\bar{p}_B}(x, y)$ is the c.d.f. of the Negative Binomial Distribution with probability of success $\bar{p}_B$, the number of successful trials x and the number of failed trials y. Parameters b and e are computed through the procedure in Section 4.2.2.1 and can be taken as the number of scenarios that can beat scenario i with probability 1, and the number of scenarios that can beat scenario i for some moderate probability, respectively. Appendix C.1 gives further details about how the equation above is set up and how $\bar{p}_B$ is computed.

[3] As the comparisons with high success probability will usually be done first, the statistical inference about $\pi_0(i')$ could be done with fewer comparisons. This simplification will cause $M_{\pi_0(i')}$ to be overestimated. This is the reason that this method is adopted here instead of in Section 4.2.2.

4.3. Fixed C Procedure

By approximating W by $\hat{W}$, we simplify the original stochastic nonlinear integer problem to the following problem:

$$\min \quad \hat{W}(k, n_0) \tag{4.10}$$

$$\text{s.t.} \quad C - kn_0 = C_1 \tag{4.11}$$

$$k \geq 40/p \tag{4.12}$$

$$n_0 \geq 30 \tag{4.13}$$

where Constraint (4.11) expresses the allocation of computational budget between the outer level and inner level, Constraint (4.12) stands for the minimal empirical requirement for k in order to keep the validity from Chapter 3.3, and constraint (4.13) tries to satisfy the normality assumption for simulated payoffs (see Lesnevski et al. (2008)). These are just some rough guidelines for large enough sample sizes to get adequate coverage of the estimated confidence interval, which is guaranteed only as $k \to \infty$ and $n_0 \to \infty$.

The quantitative property of $\hat{W}(k, n_0)$ depends on the portfolio studied. But for qualitative analysis only, there are some good properties regardless of portfolios studied. When n_0 is fixed, $\hat{W}(\cdot, n_0)$ is decreasing then increasing in k and looks like a quasi-convex function of k. When k is fixed, $\hat{W}(k, \cdot)$ is roughly increasing then decreasing and finally increasing in n_0. This is because of the two-fold function of n_0 we discussed before. Yet in reality, $\hat{W}(k, \cdot)$ usually contains many peaks and valleys due to the discreteness of n_0. These properties of $\hat{W}(k, n_0)$ require special consideration in regards to global convergence

when multivariate convex minimization techniques (such as the cyclic coordinate method (Bazaraa et al., 2005)) are adopted to solve Problem (4.10) – (4.13).

For the k coordinate searching step, the Golden Point Intersection method (Bazaraa et al., 2005) is adopted. For n_0 coordinate searching, considering the complexity of $\hat{W}(k, n_0)$, a grid search hybrid with the Armijio method (Bazaraa et al., 2005) is adopted here. The grid search method simply determines the objective value at each grid point. It may offer some protection against local minima but it is not very efficient. The Armijio method is adopted to search around the best grid point to hasten the convergence speed. The search methods we adopt here can only guarantee a local minimum. If $\hat{W}(k, n_0)$ is strictly quasi-convex, the solution is also global. Unfortunately, $\hat{W}(k, n_0)$ in general does not have such a property. The application of the grid search method helps to protect against local minima if the grid points are dense enough. Our experiments show that dozens of grid points together with the Armijio method can help to find the global minimum very quickly.

With the optimization techniques described above and the prediction procedure in Section 4.2.2, locally optimal parameters (k^*, n_0^*) are archived for the optimization of $\hat{W}(k, n_0)$. Then the "robustness" step is adopted to fine tune (k^*, n_0^*) locally so that it makes the two-level simulation work well and be stable.

The (k, n_0) tuning procedure for a given total number of simulated payoffs C is described as follows.

(1) **Initialization**

Set the minimum of n_0, $n_{0,\min}$, as 30 and set n_0^* as n_{00} and k as k_0.

Set the minimum allowed value of k, $k_{\min}$, as $40/p$ and calculate $k_{\max}$ from

Constraint (4.11) under given n_0^* .

Let $a \leftarrow 0$ and $\hat{W}_N^* \leftarrow \hat{W}(k_0, n_{00})$.

(2) **Do (with the most likely predictor $\mathcal{P}_1$):**

(a) **Searching for k^***

Golden Method to search for the optimal $k^* \in [k_{\min}, k_{\max})$ under given n_0^*.

Let $k_{\text{old}} \leftarrow k$ and $k = k^*$.

(b) **Updating n_0^***

(i) Let $n_{\text{old}} \leftarrow n_0^*$.

(ii) If $a = 0$, Generate grids of n_0:

Generate guesses for n_0 by computing $\frac{d^2 S_{ij}(n_{00})}{(\bar{X}_i(n_{00}) - \bar{X}_j(n_{00}))^2}$ for $i = \lceil kp \rceil, j = \lceil kp \rceil + 1, \lceil 1.5kp \rceil, \lceil 2kp \rceil, \ldots, \lceil 10kp \rceil, \lceil k/4 \rceil, \lceil k/2 \rceil, \lceil 3k/4 \rceil, k$;

Calculate their corresponding $\hat{W}(k^*, n_0)$;

Find the n_0^* that has the minimal $\hat{W}(k^*, n_0)$ among all grid points;

Take Armijo search along the decreasing direction of n_0^*, keep on updating n_0^* such that $\hat{W}(k^*, n_0^*)$ is the minimum.

Set $a \leftarrow 1$.

(iii) Locally optimizing n_0^*:

Perform a climbing-hill search on both sides of n_0^*, keep updating n_0^*.

(iv) Update $k_{\min}$ or $k_{\max}$:

if $n_0^* > n_{\text{old}}$, let $k_{\max} \leftarrow k^*$; else let $k_{\min} \leftarrow k^*$.

(v) Update $\hat{W}_N^* \leftarrow \hat{W}(k, n_0)$

While $(|n_{\textbf{old}} - n_0^*| > 0.1 n_{\textbf{old}} \vee 10$ **or** $|k_{\textbf{old}} - k^*| > 10/p)$.

(3) **Robustness Step (with the pessimistic predictor $\mathcal{P}_2$):**

(a) Set $\hat{W}_D = \hat{W}(k^*, n_0^*)$ with the prediction procedure in Section 4.2.2.2, $a \leftarrow 0$.

Set $k_{\min} = 40/p \vee k^*/4$ and $k_{\max} = 4k^*$.

Set $k^R = k^*$ and $n_0^R = n_0^*$.

(b) DO (with the prediction procedure in Section 4.2.2.2):

(i) Set $n_{\text{old}} \leftarrow n_0^R$ and perform a climbing-hill search on both sides of n_0^R, keep updating n_0^R.

(ii) Set $k_{\text{old}} \leftarrow k^R$ and use the Golden Method to search for the optimal $k^R \in [k_{\min}, k_{\max})$ under given n_0^R.

(iii) Update $k_{\min}$ or $k_{\max}$:

if $n_0^R > n_{\text{old}}$, let $k_{\max} \leftarrow k^R$; else let $k_{\min} \leftarrow k^R$.

(iv) Update $\hat{W}_D^* \leftarrow \hat{W}(k, n_0)$

While $(|n_{\text{old}} - n_0^R| > 0.1 n_{\text{old}} \vee 10$ or $|k_{\text{old}} - k^R| > 10/p)$ and $a < 3$.

(c) If $\hat{W}_D^R > 1.1\hat{W}_N^*$, set $k^R = k^*/1.2$ and $n_0^R = 1.2 n_0^*$.

(4) Output: (k^R, n_0^R).

4.4. Fixed T Procedure

Similarly, by approximating W by $\hat{W}$, we simplify the original problem to Problem (4.14)

$$\text{(4.14)} \quad \min \quad \hat{W}(k, n_0)$$

$$\text{(4.15)} \quad \text{s.t.} \quad C_1 = \frac{T - kn_0R_{\text{sam}} - kR_{\text{scr}} - (k\log k)(R_{\text{scrsort}} + R_{\text{com}}) - Mn_0R_{\text{scr2}}}{R_{\text{sam}}}$$

$$k \geq 40/p$$

$$n_0 \geq 30$$

where Constraint (4.15) expresses the allocation of computational time between the outer level and inner level. The other constraints have the same meaning as Constraint (4.12)–(4.13).

With other factors fixed, $\hat{W}(k, n_0)$ decreases in C_1, the total simulated payoffs at the second stage, which also decreases in k. Then $\hat{W}(\cdot, n_0)$ decreases and then increases in k and looks like a quasi-convex function. However, in most cases, $\hat{W}(k, \cdot)$ does not show any sort of convexity in n_0 due to the fact that Mn_0R_{scr2} (denoted as T_{scr2}) goes up, down and then up as n_0 increases, as shown in Figure (4.4). The effect of increasing the first-stage sample size n_0 is two-fold: on one fold, the computation time for each comparison increases as n_0 increases, and on the other fold, the total comparisons need to be done, M, may decrease as n_0 increases. Usually we have such observations: when n_0 is pretty small or relative large, the first effect dominates, then $\hat{W}(k, \cdot)$ increases in n_0; otherwise, the second effect dominates and $\hat{W}(k, \cdot)$ decreases in n_0.

The procedure when T is fixed is almost the same as the procedure described in Section 4.3. The only difference is that k^R is set to be $k^*/1.1$ in the "robustness" step when $\hat{W}_D^R > 1.1\hat{W}_N^*$.

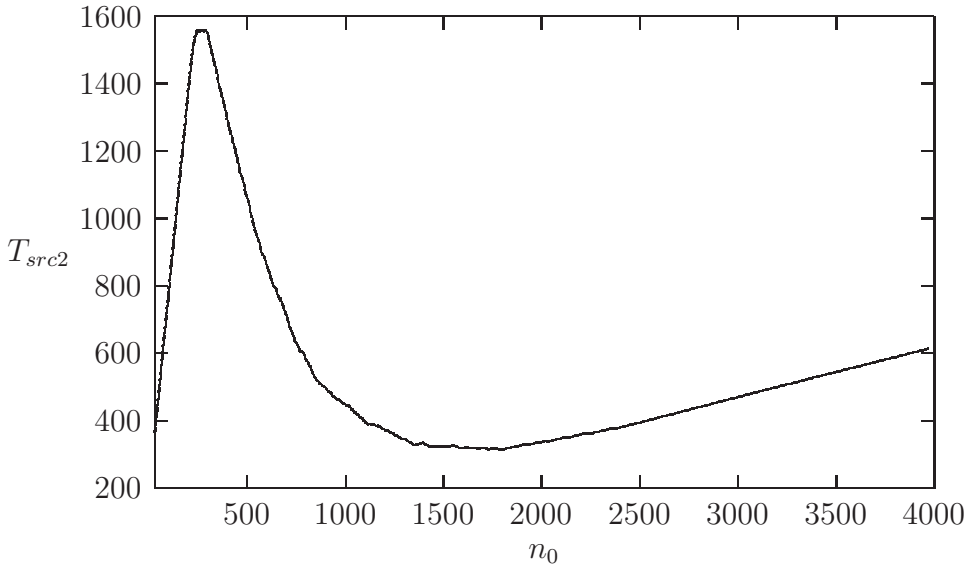

Figure 4.4. Relation of T_{scr2} and n_0

4.5. Experiments

In this section, we test the "goodness" of the pair of (k, n_0) suggested by our procedures. By "goodness" we mean that (k, n_0) is close to the optimal choice and robust to Monte Carlo variability in different runs of the two-level simulation. As the user usually does not rerun the same experiment with different random seeds to obtain the average output, it is important to have the pair (k, n_0) suggested by our procedures work robustly as well as efficiently.

The test problems we used are the experiments of selling a single put option, and holding a portfolio of options on two stocks discussed in Chapter 1. For simplicity, we call them Problem 1 and Problem 2. To enrich the test variability, we run experiments on Problem 2 by adopting different pricing models. Through this trick, we test our procedures

facing different levels of variances and covariances in the inner level of simulation. These models were introduced in Chapter 1.

We tested the simulation procedures by producing a 90% confidence interval (CI) for $\mathrm{ES}_{0.99}$ of Problem 1 and Problem 2. The $\mathrm{ES}_{0.99}$ of Problem 1 is \$3.39 and that of Problem 2 is \$32.40. Unless specified, the number of macro-replication is 20. The decomposition of the total allowed error probabilities, $\alpha = \alpha_o + \alpha_s + \alpha_{hi} + \alpha_{lo}$, is far less important to the performance of the two-level simulation than the two parameters (k, n_0), where the terms on the right side are error probabilities allocated to an outer-level confidence interval, screening, and inner-level lower and upper confidence limits, respectively. The composition of $\alpha_o = \alpha/2,\ \alpha_s \approx \alpha/4,\ \alpha_{hi} = \alpha_{lo} = (\alpha - \alpha_o - \alpha_s)/2$ works well in all our experiments.

For all the following experiments, the coverage rates of all the estimated confidence intervals of ES are greater than the nominal 90% when the experiment can be accomplished within given computational budgets.

We ran the experiments on a PC with a 2.4 GHz CPU and 4 GB memory under 64-bit Red Hat Linux. The code was written in C++ and compiled by `gcc` 3.4.6.

We set $k_0 = 40/p$ and n_{00} automatically by the procedure in Appendix C.2.1 to ensure that $\mathcal{T}(k_0, n_{00})$ is accurate enough for the prediction procedures to work well.

4.5.1. Effectiveness and Robustness of the Given C Procedure

To show the effectiveness of our procedure, we compare the estimated confidence interval width simulated with the chosen (k, n_0) of our procedure and those pairs of (k, n_0) selected

evenly from the feasible region. For Problem 1, the number of scenarios k and first-stage sample size n_0 are respectively selected from $250 \leq k \leq 128{,}000$ and $10 \leq n_0 \leq 800$ for $C = 16{,}000{,}000$. A total of 62 pairs of (k, n_0) were systematically selected, 4 more design points are selected around the best design of the 62 pairs. With some pairs of (k, n_0) like $(k, n_0) = (256{,}000, 30)$, the two-level simulation is extremely time-consuming and the confidence intervals estimated are obviously far from the best. For such a case, we only ran a small number of macro replications, such as 5 or 10. Otherwise 100 replications were tested for each pair of (k, n_0) selected. A total of 4877 macro replications were run. The selected (k, n_0) and corresponding experimental results are listed in Appendix C.3.1.1. Four typical pairs of (k, n_0) and their simulation results are shown in Figure 4.5. As displayed in the figure, within a relatively wide range of (k, n_0), the variances of the estimated confidence interval width generated by different runs of the simulation are relatively small, which means that the two-level simulation works stably with such parameters and it is both feasible and meaningful to find a good pair of (k, n_0) within such a range.

With $C = 16{,}000{,}000$, the procedure to find good parameters was run 20 times with same k_0 but different pilot simulations $\mathcal{T}(k_0, n_{00})$. For each of the 20 times our algorithm generated a pair of (k, n_0), with which 20 replications of the two-level simulation were run. The results are shown in Table 4.1. The first two columns are the pairs of (k, n_0) suggested by our procedure. The sample average and sample variance of the widths of the simulated confidence intervals are listed in the third and fourth columns. For the number of scenarios, k, the suggested value ranges from 146,908 to 171,301, changing at most 17%. The first stage sample size, n_0 varies from 65 to 70, changing at most 8%. Meanwhile,

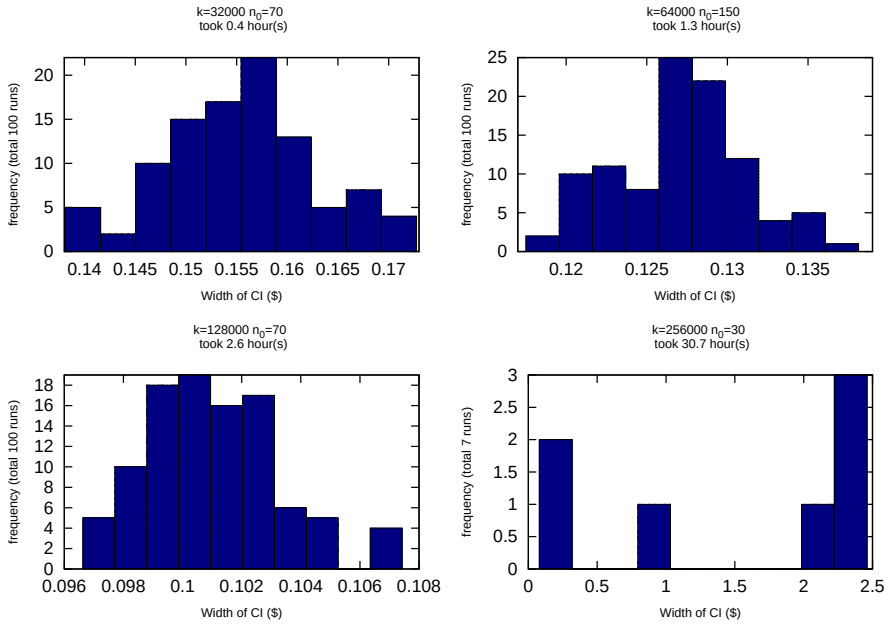

Figure 4.5. Histogram of Simulation Performance of Problem 1 with Different Pairs of (k, n_0)

the magnitudes of the sample variances are roughly 10^{-2} of those of the sample averages, which indicates that the simulation performance under the suggested pairs of (k, n_0) is very stable. The experiment shows that our procedure can find a pair of (k, n_0) with which the two-level simulation works robust, and almost as well as possible. On other hand, the pairs of (k, n_0) suggested from our procedure are stable across different pilot simulations $\mathcal{T}(k_0, n_{00})$. Figure 4.6 confirms the description above vividly. The cross points are the (k, n_0) evenly selected from the feasible region.[4] The star points are the (k, n_0) suggested by our procedure. The left plot shows that the suggested pairs of (k, n_0) cluster together. In the right plot, we note that all the star points cluster together around the left bottom corner, which means simulation with suggested (k, n_0) perform the best in

[4]To better illustrate the efficiency of the suggested pairs of (k, n_0), we also run some simulations with parameters (k, n_0) from the region $([10{,}000, 20{,}000], [50, 80])$ that seems most likely to contain the best choice(s).

k	n_0	Average CI Width	Variance of CI Width
155781	65	0.100002	1.42e-03
159774	65	0.100571	1.47e-03
155781	65	0.100190	1.45e-03
159464	65	0.100697	2.30e-03
155781	65	0.100622	1.36e-03
163465	65	0.100579	1.79e-03
146908	70	0.102005	2.10e-03
163680	65	0.100867	1.50e-03
155781	65	0.099931	1.47e-03
159292	65	0.100243	1.50e-03
155214	65	0.100557	1.81e-03
155781	65	0.100306	1.69e-03
163808	65	0.101246	1.63e-03
163503	65	0.100272	1.64e-03
155781	65	0.100013	1.63e-03
150402	70	0.102309	2.21e-03
159511	65	0.099805	1.67e-03
154999	65	0.100354	1.47e-03
155664	65	0.099898	1.68e-03
171301	65	0.102626	3e-06

Table 4.1. Experiment Results for Problem 1

terms of accuracy and robustness. Also the algorithm to suggest the (k, n_0) works robust across the different pilot simulations, $\mathcal{T}(k_0, n_{00})$, that we tried.

For a different computational budget C or Problem 2, similar experiments have been done. The simulation results are listed in Appendix C.3.1.2. All results confirm that our procedure can find efficient and robust pair of parameters (k, n_0) for the two-level simulation.

4.5.2. Effectiveness and Robustness of a Given T Procedure

The efficiency and robustness of our procedure for given computational time T can be illustrated in a similar way. We define the accomplishment rate as the ratio of the number

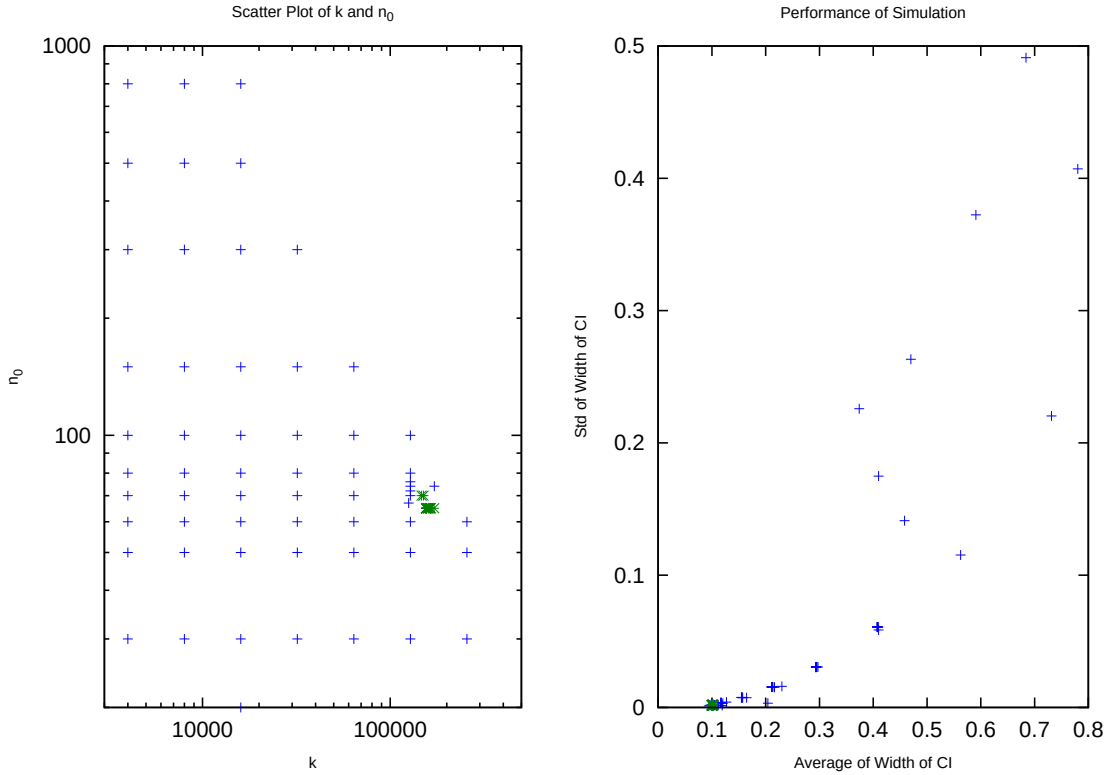

Figure 4.6. Simulation Performance of Problem 1 with (k, n_0) Suggested by our Procedure and Evenly Selected

of replications of the two-level simulation successfully accomplished within a given time T to the total number of replications we try to run. This is different from the total computational budget given in the total number of payoffs simulated C, in which case, it is natural to tell if a pair of (k, n_0) is feasible or not without any prior information about the behavior of the two-level simulation. When the total computational budget is given by clock time T, the event that a two-level simulation will complete within given time T is a random event. By assuming that the running environments remain almost the same between the two-level simulation and the test run to generate $\mathcal{T}(k_0, n_{00})$, our procedure

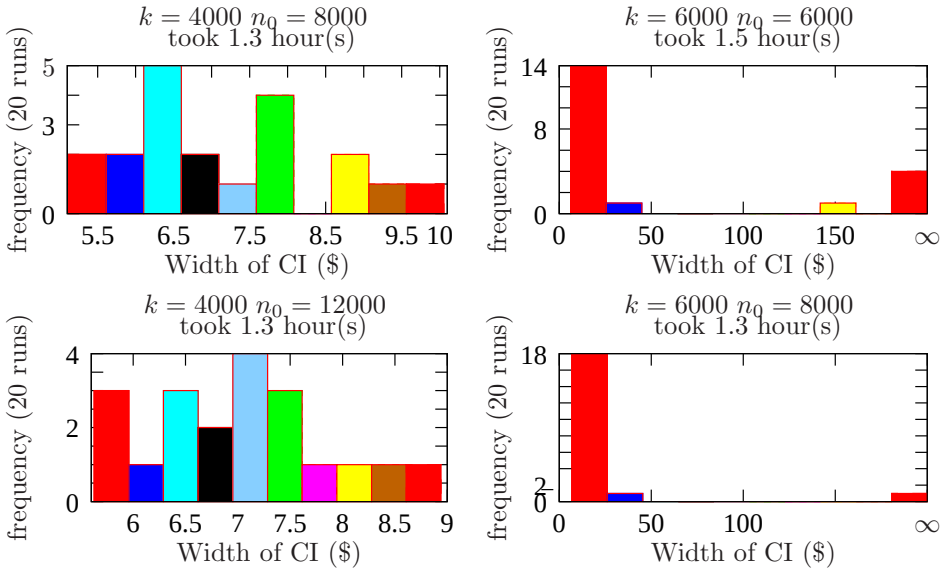

Figure 4.7. Histogram of Simulation Performance of Problem 2 with Different Pairs of (k, n_0)

can find a pair of (k, n_0) that is not only likely to be feasible, but also close to the best attainable in very similar running environments.

With $T = 240$ seconds, k and n_0 were systematically selected from $2000, 4000, 6000, 8000, 12000$ and $1000, 2000, 4000, 6000, 8000, 12000$ respectively. For each choice of settings, 100 replications of two-level simulation were run. The collective data are shown in Appendix C.3.2.2. Four typical pairs of (k, n_0) and their simulation results are shown in Figure 4.7. The experiment results indicate that there is no simple rule to choose (k, n_0) for a given T as the relationships of W and (k, n_0) are not monotone, and the two parameters, k and n_0, depend on each other. The pair, $(k, n_0) = (6000, 6000)$ is very risky and should not be suggested by our procedure, although sometimes the two-level simulation can generate very narrow confidence intervals. As shown in Figure 4.8, compared to an arbitrarily selected (k, n_0), the pair of (k, n_0) suggested by our procedure is demonstrated

to be effective and robust. The experiment was done with $T = 240$ seconds and a total of 20 pairs of (k, n_0) are found with different pilot information $\mathcal{T}(k_0, n_{00})$. Twenty replications of the two-level simulation were run to test the effectiveness and robustness of each algorithm generated pair of (k, n_0). The accomplishment rate of all the experiments with (k, n_0) suggested by our procedure is 100%. The results are also shown in Table 4.2. Similar to Table 4.1, the first two columns show the suggested pairs of (k, n_0). Compared with results of the simulation of Problem 1, the suggested pairs of (k, n_0) vary within a larger region, that is, the number of scenarios, k, changes at most 50% while the first-stage sample size, n_0, changes at most 64%. This is because of the high variance of the Monte Carlo simulation of Problem 2. But the average widths of the confidence intervals with different (k, n_0) demonstrate much less variability, around 8%. We confirm again that the (k, n_0) suggested by our procedure are almost the most efficient and robust procedure parameters; and our procedure is also robust to the Monte Carlo variability in the pilot simulation $\mathcal{T}(k_0, n_{00})$.

k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
9734	5862	3.93	1.4e-01	100%
11288	4483	3.87	9e-02	100%
9689	5094	3.86	1.2e-01	100%
11015	4514	3.96	9e-02	100%
9397	5756	4.00	1.2e-01	100%
9186	6361	4.14	1.5e-01	100%
10574	4784	3.89	7e-02	100%
10622	4676	3.91	1.0e-01	100%
10417	4837	3.92	9e-02	100%
10510	5629	4.03	8e-02	100%
10538	4486	3.94	9e-02	100%
9182	6486	4.09	1.1e-01	100%
11512	4340	3.85	1.1e-01	100%
9632	5250	3.86	7e-02	100%
10798	4671	3.92	8e-02	100%
11519	4604	3.95	9e-02	100%
7837	7051	4.06	1.1e-01	100%
11086	4553	3.90	7e-02	100%
10236	5090	3.93	5e-02	100%
11767	4289	3.91	6e-02	100%

Table 4.2. Experiment Results for Problem 2

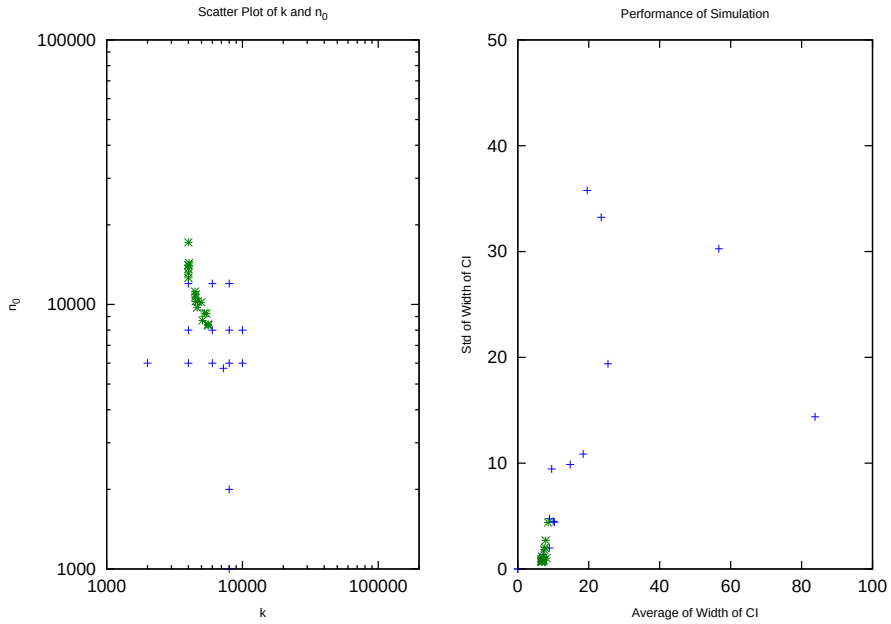

Figure 4.8. Simulation Performance of Problem 2 with (k, n_0) Suggested by our Procedure and Evenly Selected

CHAPTER 5

Parallelization on GPU

In spite of all the efficiency techniques applied, the two-level simulation is still time-consuming. Meanwhile, the development of chip-level parallelism in the microprocessor industry has already brought multicore Central Processing Units (CPUs) and manycore Graphics Processing Units (GPUs) into the mass market. It has become practicable to accelerate the two-level simulation through chip-level parallelism on regular workstations, personal computers or even laptops. In this chapter, the parallelization of two-level simulation is discussed and the experimental results demonstrate that the speedup compared with the serial implementation in Chapter 3 is applaudable.

5.1. Reason to Adopt "CPUs+GPUs"

Parallel computing has been proved to be effective to solve computationally intensive scientific problems in diverse disciplines, such as physics, geography, chemistry, biology and engineering. There has been much research on parallel computing in economics and finance in the past decade, covering topics from asset pricing to risk management. Yet the application of parallel computing in financial industry is still limited. Two obstacles exist. The first is the expensive purchase and maintenance cost of a huge or even medium-scale super computer. Second, parallel programming is more difficult than serial programming. Meanwhile, by integrating two or more processing units (cores) into one chip, chip-level parallelism is an economical solution for parallel computing. Nowadays, multicore CPUs

are popular in workstations, personal computers and laptops. The number of cores in such CPUs usually varies from 2 to 8. Typically, main memory in a computer with a multicore CPU is shared by all cores. Among different types of parallel programming languages for shared-memory parallelism, OpenMP is widely adopted for its simplicity. It provides a set of pragma clauses and interfaces to make the messy thread manipulation invisible to the programmers. Adopting OpenMP to accelerate the two-level simulation is, of course, the first choice available to us. Nevertheless, lack of scalability makes the choice less appealing when the computing resource of CPUs is not capable of handling the two-level simulation in some real challenging applications.

Most modern GPUs are fundamentally manycore processors that provide a high degree of parallelism with relatively affordable cost. Take the NVIDIA GeForce 9800 as an example. It contains 128 scalar processing elements per chip, and provides a friendly C interface package, called CUDA, through which programmers can take GPUs as additional computing resources where 128 or even more copies of C codes can be run simultaneously. The GPU is a massively multithreaded processor which creates, manages, schedules and executes threads in hardware with lightweight thread creation and zero scheduling overhead. It supports very fine-grained parallelism, allowing a low granularity decomposition of problems by assigning one thread to each data element. This property makes the parallel implementation more natural.

To address the concern of scalability and also fully utilize all computing resources of current mainstream computers, we adopt "CPUs + GPUs" in our research. The parallel implementation is on a basis of "OpenMP + CUDA".

5.2. How to Integrate GPU with CPU

The GPU organizes the massive threads in a hierarchy of thread→ block→ grid. When a piece of code, called kernel, is launched on a GPU, a grid is created. A grid may contain many blocks, and a block may contain up to 512 threads. A grid can choose to index its blocks in a 1-dimensional, 2-dimensional or 3-dimensional way. A block can also choose to index its threads in a 1-dimensional, 2-dimensional or 3-dimensional way. The thread hierarchy is for the ease of 1-to-1 mapping of threads and data elements. For example, when we generate k market scenarios, we can equip each thread with one random number generator and make it generate its portion of market scenarios. When the data processed are 2-dimensional or 3-dimensional, we can choose to index the threads in a 2-dimensional or 3-dimensional way respectively. Threads on CPU are less flexible. Each thread is assigned with an unique index. Programmers need to maintain the mapping between threads and their tasks on their own.

Another characteristics of GPU is the visibility of different types of memories. Unlike a CPU, a GPU exposes the manipulation of different types of on-chip memories to programmers. Each thread may have its own local storage: registers. Threads within a block can communicate with each other through shared memory which is up to 16K per block and as fast as registers. All threads can access the global memory on the chip, which is larger yet much slower than shared memory and registers. There are some other types of memory, such as texture and constant memory, similar to the global memory yet cached for optimized local access. Compared with the beauty of thread manipulation, use of memory is complex and less efficient due to the data bus limit. Because of these characteristics of GPU, it is good at data-light massive-thread parallelism and becomes

less appealing when data accessing happens frequently. However, for most CPUs, there is on-chip L1/L2 cached memory together with the main memory of the computer. Use of different types of memories is optimized and invisible for programmers, which makes the serial implementation on CPUs easy. But such cache mechanism may bring race conditions for shared memory parallel implementation. Data racing is one of the most often-seen race conditions, which happens when a slot of shared memory is intended to be updated by different threads with different contents. For example, as each thread may have its own copy of the shared memory, two or more threads may write simultaneously to their own local copies and then update the shared memory whenever necessary. Conflict happens during the updates when local copies are different. To reduce data racing condition, special techniques need to be adopted. This breaks the simplicity of cached memory and makes the shared memory parallel implementation on GPU complicated.

Obviously, the parallelism of GPU is similar to, and to some extent even simpler than, shared memory programming on CPUs due to the fact that data racing becomes easier to handle as the memory of GPU is usually not cached. Yet because of the limited amount of on-chip fast registers and shared memory, a large amount of data has to be stored in the global memory with much longer latency. To achieve good performance, the communication between shared memory (or registers) and global memory needs to be optimized and the data reuse on shared memory (or registers) should be maximized, which is very similar to distributed memory programming.[1] In other words, it is easy to parallelize on GPU but there is much more work to make it really efficient. Fortunately, there are

[1]Distributed-memory super computers are most widely deployed due to the scalability. Each processor has its own memory and connects to each other through high speed networks. The communication between the processors is usually the bottleneck of the performance.

many examples in the field of shared memory and distributed memory programming from where we can borrow ideas on how to efficiently reduce communication and reuse data to improve the performance.

The two-level simulation can be taken as a pipeline consisting of the following steps: generating k market scenarios, first-stage sampling of payoffs, screening, second-stage sampling of payoffs and computing the confidence interval. In reality, sampling payoffs and generating market scenarios, are typically the most time-consuming. Fortunately, those steps are "embarrassingly parallel" problems which can be easily parallelized with linear speedup with respect to the number of parallel processors used. In some simplified cases, like Problem 1 in Section 1.1, or when a large amount of computing resources is allocated, the time spent on screening and computing the confidence interval is not negligible any more. Then these two steps also need to be parallelized for an effective implementation. The time-consuming part of screening, as demonstrated in Section 3.2.1, is to compute the variances of the differences of payoffs simulated conditional on some selected pair of market scenarios. The selection of pairs is adaptive based on the nondecreasing order of the estimated values of the portfolio in each scenario. Therefore, synchronization of the concurrent comparisons and data communication are required to achieve good performance. The computation of the confidence interval relies on finding the root of a group of nonlinear equations; see Baysal and Staum (2008). The data dependency between each iteration of root finding algorithm brings difficulty to parallel implementation. We take the Problem 1 in Section 1.1 as a representative to demonstrate how to integrate GPUs with CPUs together for sophisticated simulations in financial engineering.

5.2.1. Generating Scenarios and Conditional Payoff Sampling

Generating market scenarios and sampling payoffs are typical of Monte Carlo simulation, which is "embarrassingly parallel". In this section, parallelism of such Monte Carlo simulation on multiple GPUs is discussed and some load balancing techniques are adopted to improve the efficiency.

5.2.1.1. Random Number Generator. A general approach for implementing a random number generator for parallel computing is to partition the sequence of a long period random number generator into disjoint subsequences and use a different subsequence on each computing node; see L'Ecuyer (1998). We use the random number generator proposed by L'Ecuyer (1996), implemented in 32-bit integer arithmetic with splitting facilities of L'Ecuyer and Touzin (2000) to generate uniform variates. For normal variates, the Box-Muller transform is adopted according to the suggestion of Thomas et al. (2009).

5.2.1.2. Generating Scenarios and the First-stage Sampling. There are two stages of sampling in the two-level simulation. Usually CRNs are adopted to reduce the variance in the first stage, and then the storage of each conditional payoff is necessary. For the second-stage sampling, only the conditional sample means and sample variances are important. Because of this difference, each stage of sampling has its own implementation.

The GPU can support, and actually expects, thousands of concurrent threads to hide the latency that may occur when accessing the global memory. The number of market scenarios, k, may vary from thousands to millions depending on the total computational budget. The number of threads launched on each GPU is fixed, denoted as N in our design. Each thread is equipped with a random number generator and deals with $\frac{k}{NG}$ market scenarios, where G is the number of GPUs. The number of threads N is set to

4096 and G is 4 by default.[2] The sampling of market scenarios and the first-stage sampling of payoffs are combined here to avoid spending time on accessing market scenarios on the global memory of each GPU. The sampled payoffs have to be stored in the global memory because no other memory can accommodate them and the global memory is the only channel to communicate with the CPUs. For speed of memory access on CPUs, which is automatically locally cached, we store all the conditional payoffs of a market scenario in a contiguous linear memory.

An outline of the algorithm for generating market scenarios and the first-stage conditional payoffs is as follows.

CPU's part:

(1) Create G CPU threads, each having a unique GPU as its additional computing resource. Let g be the thread ID, starting from 0, and also the index of GPUs. [3]

(2) Each CPU thread g computes sequentially N random number seeds starting from $(gN + 1)$th stream. Each seed points to the beginning of a unique stream. A stream is a split of the whole sequence of random numbers (see L'Ecuyer (1998)). Each CPU thread shares the seed of the 0th stream which is taken as the common random number seed on all CPU threads and feeds to its own assigned GPU the N random number seeds.

GPU's part:

(1) Let $i \leftarrow$GPU thread ID, a distinct integer from $[0, \cdots, N-1]$.

[2]In reality, two or more GPUs are encapsulated in one chip. In our context, GPU stands for a conceptual Graphics Processing Unit, not a chip. So does CPU.

[3]There is no necessity to have G CPUs on board, yet if CPU threads take part in time-consuming computing, it is better to have more than $G/2$ CPUs on board to avoid CPU threads waiting for CPU time.

(2) While $i < \frac{k}{G}$

(a) Read the $(i\%N)$th random number seed from the global memory, where $i\%N$ stands for the remainder of i/N.[4]

(b) Generate market scenario i from the random number generator.

(c) Save the state of random number back to the $(i\%N)$th seed on the global memory of the GPU.

(d) Read the set of common random seed.

(e) Generate n_0 payoffs conditional on market scenario i using the common random seed, and store them contiguously on the global memory.

(f) $i \leftarrow i + N$.

5.2.1.3. The Second-stage Sampling. The second-stage sampling is pretty similar to the first stage sampling described above, except that thread private[5] random number seeds are adopted instead of common seeds and the sample mean and sample variance are computed instead of storing all sampled payoffs on the global memory on the GPU. Another issue that needs to be specially addressed is the load-balance of threads on GPUs especially when the number of market scenarios retained after screening, $|I|$, is less than the total number of threads NG. Basically, in that case we assign multiple threads to a market scenario in the set I instead of the multiple-scenario-one-thread model in the first stage sampling.

Figure 5.1 demonstrates the policy we adopted here. Each box stands for the number of payoffs to simulate for the corresponding market scenario. Suppose $|I| = 400$, and 100

[4]The $(i\%N)$th random number seed on GPU g copies initial value from the $gN+(i\%N)$th random number seed on the CPUs.

[5]A slot of memory that is accessible only by its owner thread is thread private.

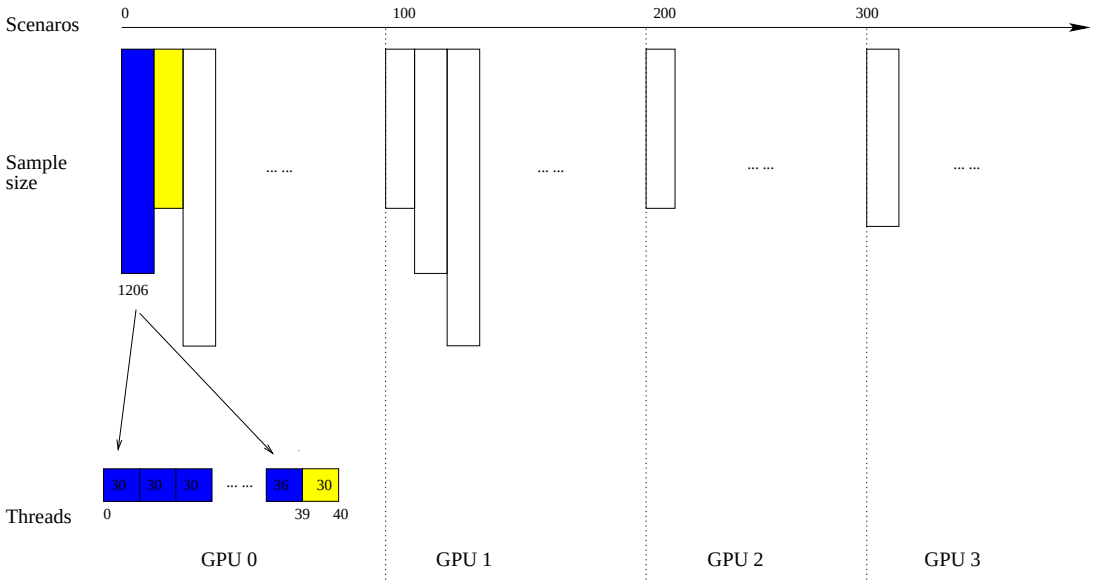

Figure 5.1. Thread Balance when $|I| = 400$, $N = 4096$ and $G = 4$

contiguous scenarios are assigned to each GPU. As an example, suppose the number of payoffs that need to be simulated on GPU 0 is 122,880 and the number of GPU threads is 4,000. Then each thread is supposed to simulate about 30 payoffs in order to keep the load balanced among the GPU threads. Suppose that 1206 payoffs need to be simulated for scenario 0. In this case, 40 threads are assigned to scenario 0, and each simulates 30 payoffs except the last one simulates 36 payoffs. [6] Through this way the work load is split almost evenly among all invoked threads. That makes the procedure well parallelized.

5.2.2. Parallel Screening

Two characteristics of the screening algorithm need to be mentioned here. First, the complexity of the screening algorithm in Chapter 3 is $\mathcal{O}(k^2)$. When k becomes larger, the

[6]Such alignments happen frequently in parallel computing. In this thesis, we simplify the discussion by not explicitly mentioning it repeatedly.

time spent on screening increases much faster than that for market scenarios or payoffs which usually share complexity of $\mathcal{O}(k)$. Thus it is necessary to implement the screening algorithm in parallel. Second, the screening algorithm is a data-heavy algorithm. Thus, the use of memory needs to be specially addressed. To reduce data communication, we divide screening into two parts: "GPU screening" and "CPU screening". In "GPU screening", the screening operation is performed on subsets of the k scenarios. Each GPU performs screening on a subset containing k/G scenarios. Hopefully a large number of inferior market scenarios are screened out. Then the surviving scenarios and their payoffs are compiled from each GPU to the main memory, and the "CPU screening" is applied on the remaining scenarios. Compared with the algorithm in Chapter 3, the set of scenarios retained after the parallel screening is a super set of that retained after serial screening, due to the fact that early elimination of scenarios in "GPU screening" may help some scenarios survive "CPU screening". Yet in our experiments, the sets of scenarios retained by the two different screening methods are always the same.

The screening algorithm contains two main steps: sorting and multiple comparisons. Sorting needs to be performed before multiple comparisons to enable the adaptive selection of the compared pairs of scenarios. Speedup of sorting on a GPU compared with a CPU is not significant because sorting is a data-heavy operation. When sorting on a GPU, the bandwidth of GPU memory is the bottleneck. Satish et al. (2009) reported 23% faster sorting on a GPU compared to a CPU. Sintorn and Assarsson (2008) claimed $6-14$ times faster when data containing only keys are sorted on GPU compared to a CPU. Yet our experience with the same algorithm applied to sorting key-value pairs shows less benefit

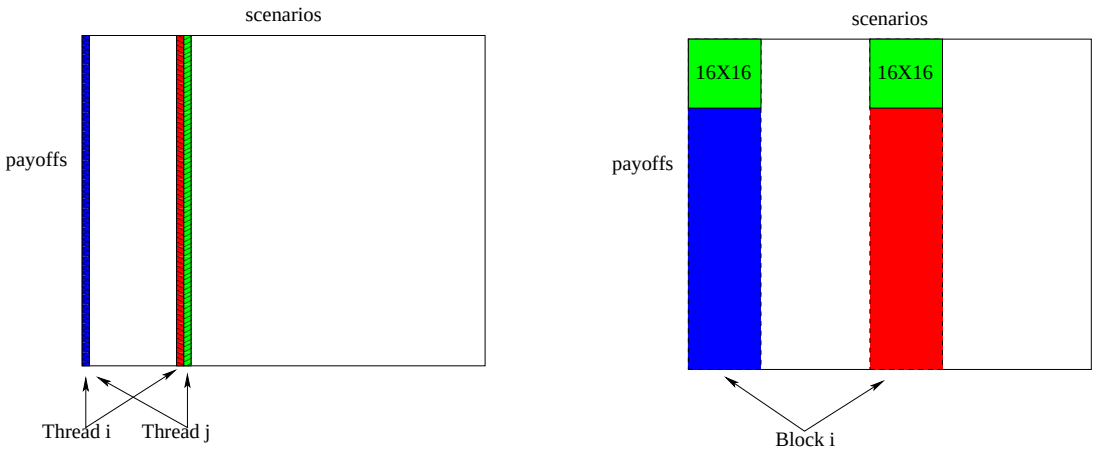

Figure 5.2. Naive and Group Comparison Parallelism

for doing sorting on GPU. So all sorting operations are performed on CPUs and quick sort is applied.

For parallelism on a GPU, a naive idea is to assign each thread a comparison. This method is not efficient because the payoffs of each scenario need to be read repeatedly by different GPU threads, as shown in the left plot of Figure 5.2. Each thin column stands for the payoffs simulated conditional on one market scenario. In this figure, thread i needs to read two columns of payoffs from the global memory. Thread j does the same operation again even through they share one column of payoffs. In this implementation, the shared data cannot be reused because either thread i and j may not be in the same block[7] or the fast shared memory may not be able to accommodate the whole column of payoffs or both.

This type of problem is often seen in distributed memory programming. Data reuse is the key to improve the efficiency. We assign a group of comparisons to a block of threads,

[7]Only threads with a block can access the same shared memory. Each block can contains no more than 521 threads.

so that the data read into the shared memory can be reused to save computational time. This idea, called group comparison, is shown in the right plot of Figure 5.2. In this figure, each column still stands for the payoffs simulated conditional on one market scenario. Initially, a block with 256 threads reads two 16×16 sub-matrices of the payoffs into the shared memory.[8] The sample variance of the 16 differences is computed simultaneously in each thread. Then, the block moves on to read the next two 16×16 sub-matrices of payoffs until the sample variance of all n_0 differences are computed and comparisons are done. For example, suppose the pairs of scenarios (i, j) for $i = 0, \ldots, 15; j = 32, \ldots, 47$ need to be compared and 256 threads are created for this purpose. According to the naive method, each thread needs to read $2n_0$ payoffs from the global memory. Global memory accesses need to be performed 512 times in total. By the group comparison we proposed here, each thread needs to read $\frac{n_0}{8}$ payoffs. In total, $2n_0$ memory accesses are required due to the fact that 16 memory accesses performed on 16 contiguous threads are coalesced into one access in our setting. [9] For convenience, we require n_0 be a multiplier of 16.

The outline of parallel screening is shown in Figure 5.3. In the figure, Thread 0 to 3 are the threads on CPUs, each invokes one GPU as an additional computing resource. The dashed lines enclose the piece of procedure running on a GPU. There are 4096 threads running on a GPU simultaneously. As indicated in the figure, there are interactions between the master CPU and its slave GPU during the "GPU screening", because we

[8]Each thread only reads 2 conditional payoffs, and the index of thread in a block can be mapped into the index of payoffs easily.

[9]On a GPU, if all threads in a half-wrap access "element size" the contiguous memory starting at "16 $\times$ element size", the 16 memory accesses may be coalesced into one access. The "element size" could be 4, 8 and 16 bytes.

want to keep to chance to terminate early before we compare all possible pairs on a GPU. This strategy proves to be efficient.

5.3. Computing the Confidence Interval

Baysal and Staum (2008) discuss how to compute empirical likelihood confidence interval for expected shortfall. We adopt the bisection method to compute the confidence interval in Section 3.2.1, as the Newton method may not converge to the roots we want.

Considering that computing of the confidence lower bound (or upper bound) is the process of finding the minimum (or maximum) of several candidates, we adopt OpenMP to create G threads on the CPUs to compute the minimum (or maximum) of each candidate. To find the minimum (or maximum) of a candidate, we adopt the p-section method instead of bisection to find a root of the likelihood function, where p is number of sub-intervals partitioned from the feasible region. We assign each sub-interval to a block of 256 threads. All threads within a block work together to evaluate the likelihood function value at the endpoint of the sub-interval. For the interval under consideration, a total of p points are tested. Two points are selected as the new endpoints of the new interval which has width $1/p$ of the previous interval.

5.4. Experiments

We ran the experiments on a PC with a 3.0 GHz Core 2 Duo CPU E8400 and 6 GB memory under 64-bit Ubuntu together with 4 GeForce 9800 GPUs. The code was written in C/C++ and compiled by `gcc` 4.2.4.

Unless specified, all experimental settings are the same as the experiments of Problem 1 in Chapter 3.3. We ran 10 macro replications for each setting. We compared the time

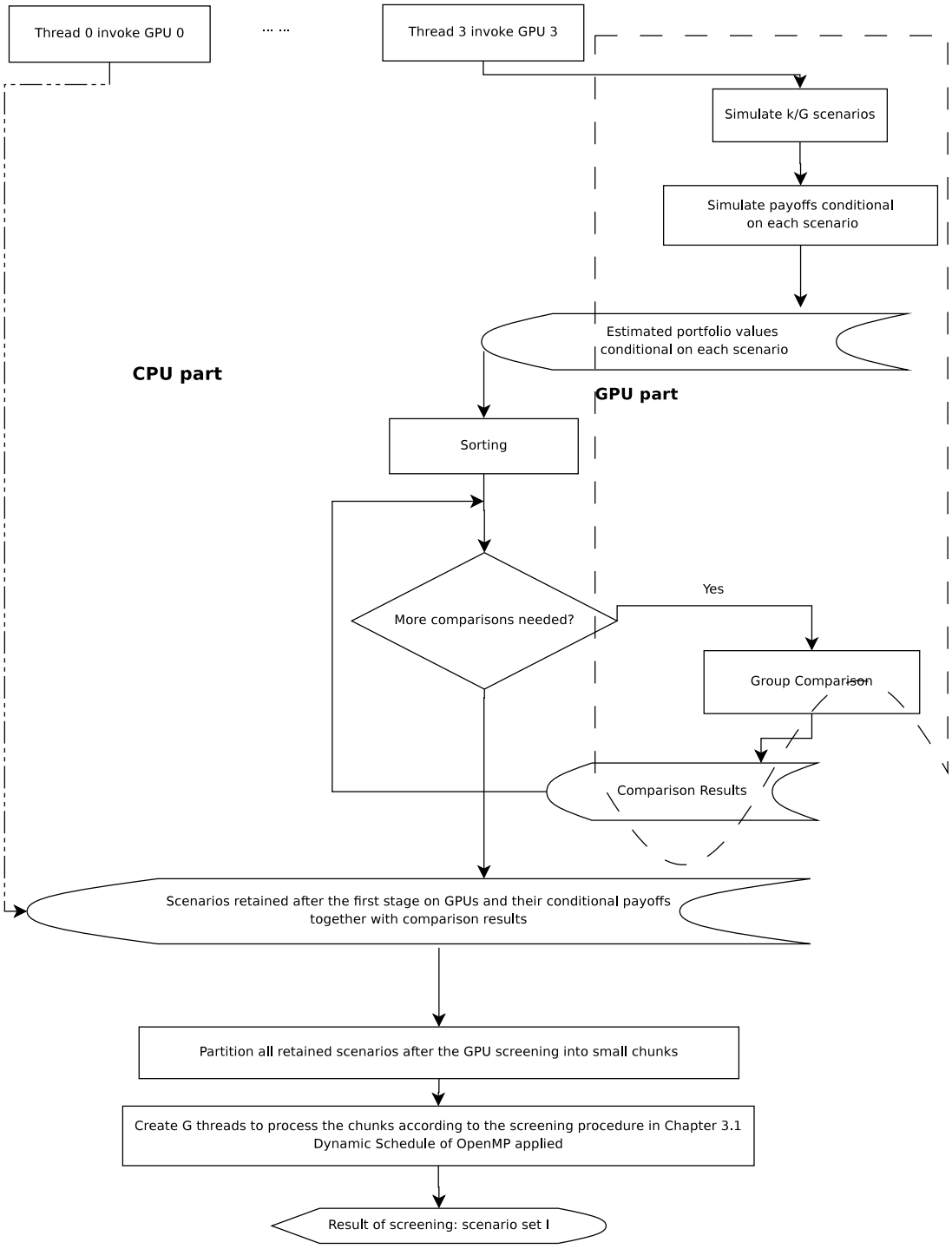

Figure 5.3. Flow of Generating Scenarios, First-stage Sampling and Parallel Screening

C	k	n_0	kn_0	$\|I\|$	C_1
2M	21999	48	1.056M	249	0.944M
4M	42265	48	2.029M	463	1.971M
8M	68013	64	4.353M	731	3.647M
16M	125262	64	8.017M	1332	7.983M
32M	214000	80	17.12M	2230	14.88M
64M	359995	80	28.8M	3717	35.2M

Table 5.1. Configurations and Internal Variables of Experiments

consumed by the parallel procedure we propose here and with the serial procedure in Chapter 3.2.1. The total number of simulated payoffs C, number of market scenarios k and the first stage sample size n_0 for the experiments are shown in Table 5.1. Some internal variables are also included as they indicate work load in each step of the procedure.

We discuss the parallel implementation for the five time-consuming steps of the two-level simulation. Among them, generating k market scenarios, the first-stage sampling and the second-stage sampling are "embarrassingly parallel." The speedups of parallel implementation on "CPUs+GPUs" of the three steps are most notable. Figure 5.4 shows the running time of the second-stage sampling of the parallel and serial procedures. It shows that the speedup of the Monte Carlo simulation ranges from 73 to 457 in our experiments, increasing with the number of simulated payoffs in the second stage.

On the other hand, screening and computing the confidence interval are difficult to parallelize due to data dependency in the algorithms. For screening, we reduce the heavy data communication by reusing fast shared memory. The speedup varies from 18 to 65, increasing with the problem size kn_0. Yet, for computing the confidence interval, the speedup by introducing p-section on GPU and multiple threads on CPU is limited. Figure 5.5 demonstrates the difference of running time for computing the confidence

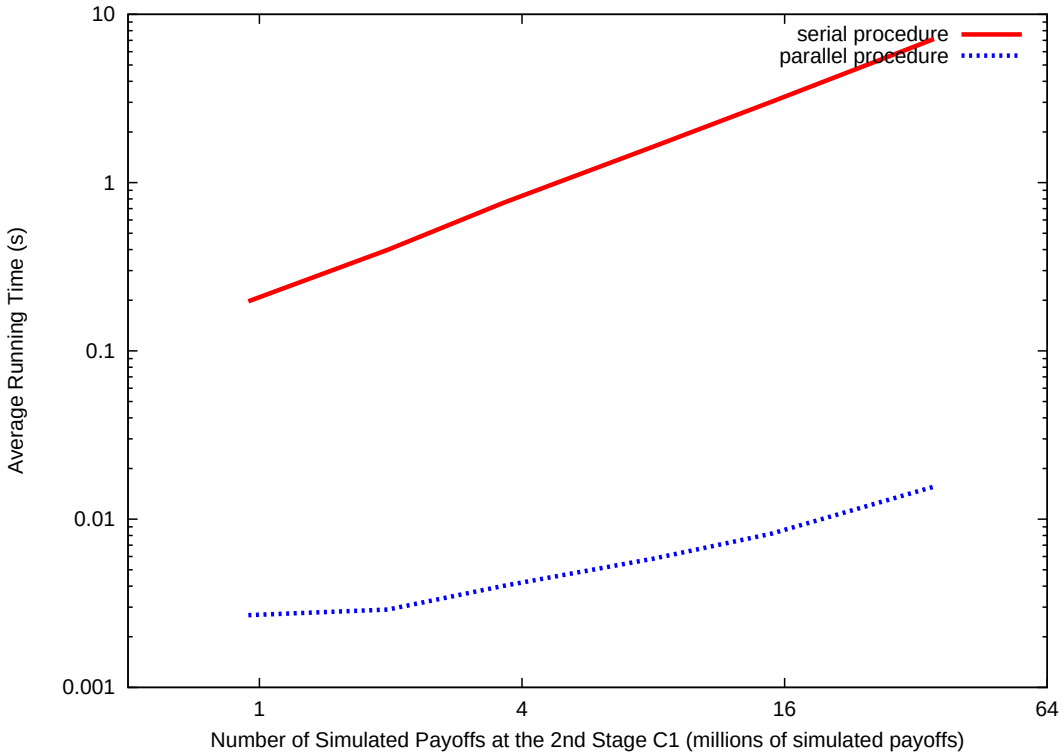

Figure 5.4. Time Spent on the 2nd Stage Sampling

interval of the parallel and serial procedures. The speedup is from 1.5 to 12, increasing as the number of scenarios retained after screening, $|I|$, increases. The speedup of computing the confidence interval is not so appealing compared with the speedup of Monte Carlo simulation. One of the reasons is the speedup of p-section over bisection is only $\log_2 p$. We have not yet found any viable alternatives due to heavy data dependency. The good news is that the fraction of the overall running time used to compute the confidence interval is less than 8% even for the worst case in the parallel implementation. It is not really necessary to worry about the speedup of this part.

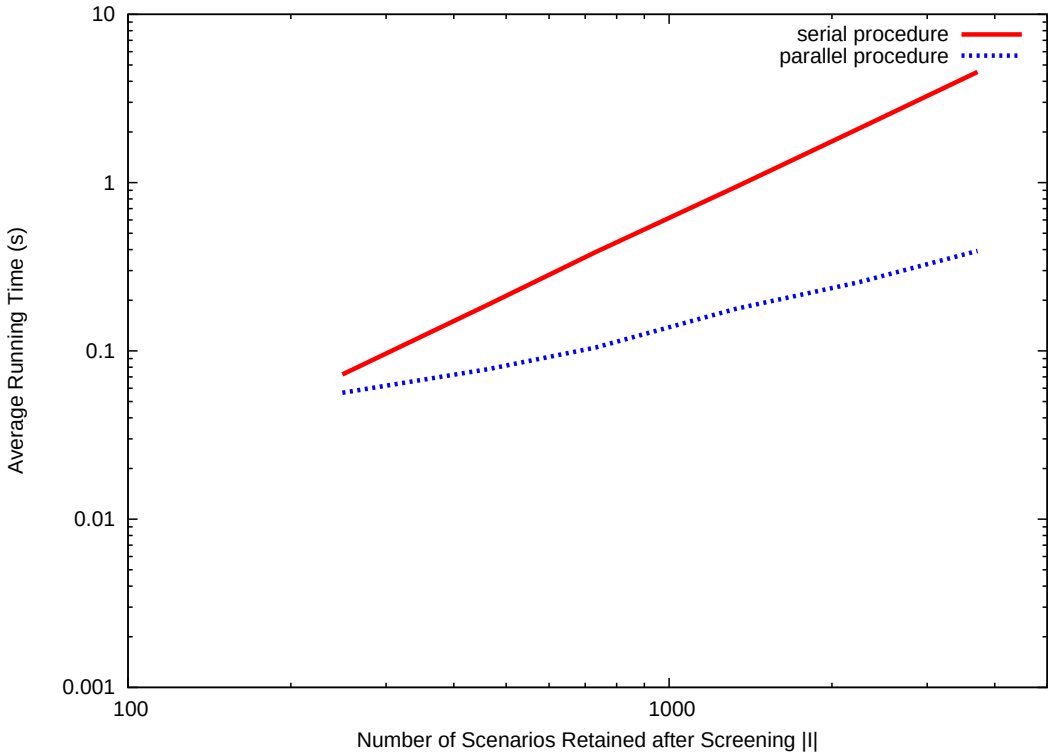

Figure 5.5. Time Spent on Computing the Confidence Interval

Overall, the speedup of the parallel procedure relative to the serial procedure is shown in Figure 5.6. The speedup ranges from 10 to 70, increasing as the total computational budget C becomes larger. For a real problem where sampling payoffs is dominant in computational time, the parallel procedure would become even better because, as demonstrated in Figure 5.4, the Monte Carlo simulation is really good for parallelism.

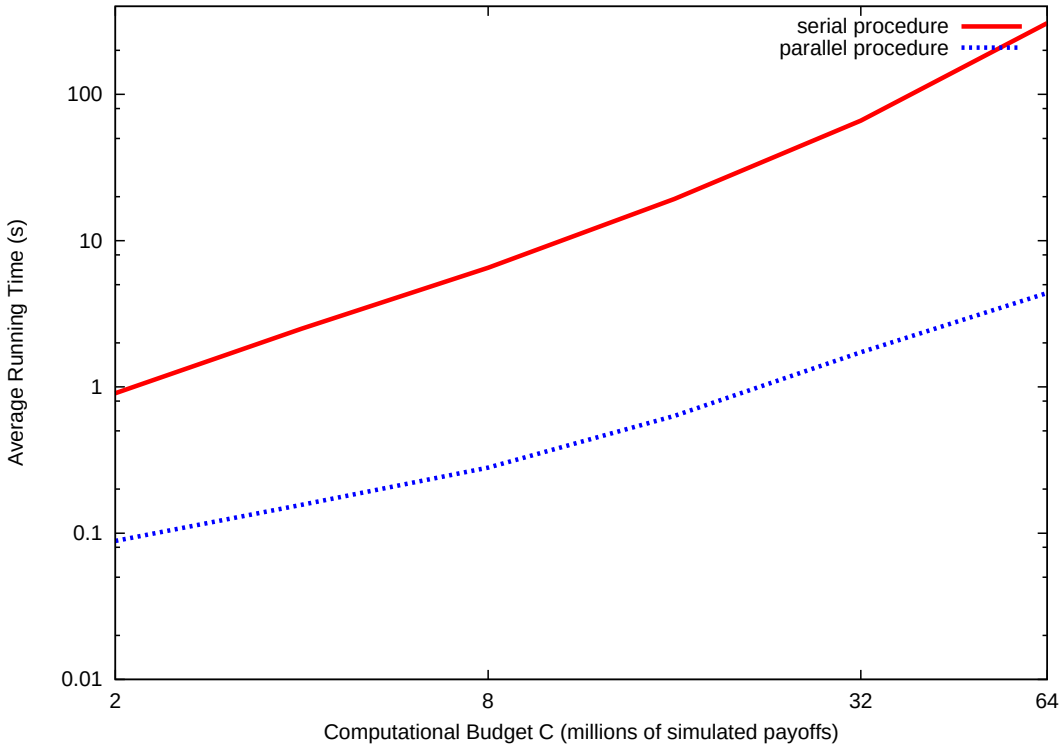

Figure 5.6. Running Time of the Parallel and Serial Procedures

References

Abbas-Turki, L. and Vialle, S. (2009). European option pricing on a GPU cluster. Technical report, ANR Supelec.

Acerbi, C. and Tasche, D. (2002). On the coherence of expected shortfall. *Journal of Banking and Finance*, 26:1487–1503.

Banerjee, S. (1961). On confidence interval for two-means problem based on separate estimates of variances and tabulated values of t-table. *Sankhyā*, A23:359–378.

Baysal, R. E. and Staum, J. (2008). Empirical likelihood for value at risk and expected shortfall. *Journal of Risk*, 11(1):3–32.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. (2005). *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, 3rd edition.

Boesel, J., Nelson, B. L., and Kim, S. (2003). Using ranking and selection to 'clean up' after simulation optimization. *Operations Research*, 51:814–825.

Chen, C.-H., Lin, J., Yücesan, E., and Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 10(3):251–270.

Chen, S. X. (2008). Nonparametric estimation of expected shortfall. *Journal of Financial Econometrics*, 6(1):87–107.

Chinchilla, F., Gamblin, T., Sommervoll, M., and Prins, J. F. (2004). Parallel n-body simulation using GPUs. Technical report, Department of Computer Science, University

of North Carolina at Chapel Hill.

Derman, E. (1999). Regimes of volatility. *Risk*, 12(4):55–59.

Fan, Z., Qiu, F., Kaufman, A., and Yoakum-Stover, S. (2004). GPU cluster for high performance computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 47, Washington, DC, USA. IEEE Computer Society.

Gordy, M. and Juneja, S. (2006). Efficient simulation for risk measurement in portfolio of CDOs. In Perrone, L. F., Lawson, B., Liu, J., and Wieland, F. P., editors, *Proceeedings of the 2006 Winter Simulation Conference*, pages 749–756. IEEE Press.

Gordy, M. B. and Juneja, S. (2008). Nested simulation in portfolio risk measurement. Finance and Economics Discussion Series 2008-21, Federal Reserve Board.

Lan, H., Nelson, B. L., and Staum, J. (2007a). A confidence interval for tail conditional expection via two-level simulation. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Proceeedings of the 2007 Winter Simulation Conference*, pages 949–957. IEEE Press.

Lan, H., Nelson, B. L., and Staum, J. (2007b). Two-level simulations for risk management. In Chick, S., Chen, C.-H., Henderson, S., and Yücesan, E., editors, *Proceedings of the 2007 INFORMS Simulation Society Research Workshop*, pages 102–107.

Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling and Analysis*. McGraw-Hill, New York, 3rd edition.

L'Ecuyer, P. (1996). Combined multiple recursive random number generators. *Operations Research*, 44(5):816–822.

L'Ecuyer, P. (1998). *Handbook on Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter 4: Random Number Generation, pages 93–137. Wiley.

L'Ecuyer, P. and Touzin, R. (2000). Fast combined multiple recursive generators with multipliers of the form a = $\pm 2^q \pm 2^r$. In *WSC '00: Proceedings of the 32nd Conference on Winter Simulation*, pages 683–689, San Diego, CA, USA. Society for Computer Simulation International.

Lee, S.-H. (1998). *Monte Carlo computation of conditional expectation quantiles*. PhD thesis, Stanford University, Stanford, CA, USA.

Lesnevski, V., Nelson, B. L., and Staum, J. (2008). An adaptive procedure for estimating coherence risk measures based on generalized scenarios. *Journal of Computational Finance*, 11(4):1–31.

Lesnevski, V., Neslson, B. L., and Staum, J. (2007). Simulation of coherent risk measures based on generalized scenarios. *Management Science*, 53(11):1756–1769.

Luenberger, D. G. (1998). *Investment Science*. Oxford University Press, New York.

McNeil, A. J., Frey, R., and Embrechts, P. (2005). *Quantitative Risk Management*. Princeton University Press, Princeton, New Jersey.

Rinott, Y. (1978). On two-stage selection procedures and related probability-inequalities. *Communications in Statistics - Theory and Methods*, A8:799–811.

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press.

Satish, N., Harris, M., and Garland, M. (2009). Designing efficient sorting algorithms for manycore GPUs. In *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Singla, N., Hall, M., Shands, B., and Chamberlain, R. D. (2008). Financial Monte Carlo simulation on architecturally diverse systems. In *Proc. of Workshop on High Performance Computational Finance*, pages 1–7.

Sintorn, E. and Assarsson, U. (2008). Fast parallel GPU-sorting using a hybrid algorithm. *Journal of Parallel and Distributed Computing*, 68(10):1381–1388.

Steckley, S. G. and Henderson, S. G. (2003). Simulation input modeling: A kernel approach to estimating the density of a conditional expectation. In Chick, S. E., Sanchez, P. J., Ferrin, D. M., and Morrice, D. J., editors, *Proceeedings of the 2003 Winter Simulation Conference*, pages 383–391. IEEE Press.

Stein, C. (1945). A two-sample test for a linear hypothesis whose power is independent of the variance. *Annals of Mathematical Statistics*, 16(3):243–258.

Sundaram, R. K. (1996). *A First Course in Optimization Theory*. Cambridge University Press.

Surkov, V. (2007). Parallel option pricing with Fourier space time-stepping method on graphics processing units. Technical report, University of Toronto - Department of Computer Science.

Tezuka, S., Murata, H., Tanaka, S., and Yumae, S. (2005). Monte Carlo grid for financial risk management. *Future Generation Computer Systems*, 21(5):811 – 821.

Thomas, D. B., Howes, L., and Luk, W. (2009). A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation. In *Proceeding of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 63–72.

APPENDIX A

Computation of the Maximum in Equation (3.8)

By the definition of $\mathbf{w}'$ in Equation (2.2), the computation of $\Delta(\ell) := \sqrt{\max_{\mathbf{w}\in\mathcal{S}(\ell)} \sum_{i=1}^{\ell}(w_i')^2}$ in Equation (3.8) is equivalent to maximizing $\sum_{i=1}^{\ell} w_i^2/p^2$ over the set:

$$\mathcal{S}(\ell) := \left\{ \mathbf{w} : \mathbf{w} \geq 0,\ \sum_{i=1}^{k} w_i = 1,\ \sum_{i=1}^{\ell} w_i = p,\ \prod_{i=1}^{k} w_i \geq c\, k^{-k} \right\}.$$

Choosing $w_{\ell+1} = w_{\ell+2} = \cdots = w_k = (1-p)/(k-\ell)$ has no effect on the objective and leads to the loosest possible constraint on $w_1, w_2, \ldots, w_\ell$, namely $\prod_{i=1}^{\ell} w_i \geq ck^{-k}((1-p)/(k-\ell))^{-(k-\ell)}$. Thus, letting $x_i = w_i/p$ for $i = 1, 2, \ldots, \ell$, we have reduced the problem to the ℓ-dimensional problem

$$\max f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{x} \in \mathcal{X} := \{\mathbf{x} : \mathbf{x} \geq 0,\ g(\mathbf{x}) \geq 0,\ h(\mathbf{x}) = 0\}$$

where $f(\mathbf{x}) = \sum_{i=1}^{\ell} x_i^2$, $g(\mathbf{x}) = \sum_{i=1}^{\ell} \log x_i + \ell \log p - \log c + k \log k - (k-\ell)(\log(k-\ell) + \log(1-p))$ and $h(\mathbf{x}) = \sum_{i=1}^{\ell} x_i - 1$. The function f is convex and the feasible set $\mathcal{X}$ is non-empty, closed, bounded, and convex. Therefore the maximum is attained at an extreme point of $\mathcal{X}$ (Rockafellar, 1970, Cor. 32.3.2).

We next turn to necessary conditions on the gradients ∇f, ∇g, and ∇h at any point $\mathbf{x}^*$ where the maximum is attained. The gradients of the objective and of the active constraints are given by $(\nabla f(\mathbf{x}))_i = 2x_i$, $(\nabla g(\mathbf{x}))_i = 1/x_i$, and $(\nabla h(\mathbf{x}))_i = 1$. The constraint $\mathbf{x} \geq 0$ is not active at any $\mathbf{x} \in \mathcal{X}$ because having $x_i = 0$ for any i leads to a

violation of the constraint $g(\mathbf{x}) \leq 0$. Because $\mathbf{x}^*$ is an extreme point of $\mathcal{X}$, its coordinates are not all equal, so the rank of $[\nabla g(\mathbf{x})\ \nabla h(\mathbf{x})]$ is two. Because this equals the number of active constraints, there exist $\lambda \geq 0$ and $\mu \geq 0$ such that $\nabla f(\mathbf{x}^*)+\lambda\nabla g(\mathbf{x}^*)+\mu\nabla h(\mathbf{x}^*) = 0$ (Sundaram, 1996, Thm. 6.10). That is, $2x_i^* + \lambda/x_i^* + \mu = 0$ for all $i = 1, 2, \ldots, \ell$. This equation has at most two positive real roots. Therefore any point $\mathbf{x}^*$ where the maximum is attained is an extreme point of $\mathcal{X}$ with the property that the set $\{x_1^*, x_2^*, \ldots, x_\ell^*\}$ contains at most two distinct values.

The only remaining questions are what those two values are and how many of the coordinates $x_1^*, x_2^*, \ldots, x_\ell^*$ take on each of the two values—by symmetry, it does not matter how the coordinates are permuted. Consider an extreme point $\mathbf{x}^{(m)}$ of $\mathcal{X}$ such that m coordinates take on one value, $a^{(m)}$, while the other $\ell - m$ coordinates take on another value, $b^{(m)}$. Because the order does not matter, we take $x_1^{(m)} = x_2^{(m)} = \cdots = x_m^{(m)} = a^{(m)}$ while $x_{m+1}^{(m)} = x_{m+2}^{(m)} = \cdots x_\ell^{(m)} = b^{(m)}$. To satisfy the constraint $h(\mathbf{x}^{(m)}) = 0$, we must have $b^{(m)} = (p - ma^{(m)})/(\ell - m)$. For each $m = 1, 2, \ldots, \ell - 1$, we compute $a^{(m)}$ by solving $g(\mathbf{x}^{(m)}) = 0$. Finally, we compute $\Delta(\ell) = \max\{f(\mathbf{x}^{(m)}) : m = 1, 2, \ldots, \ell - 1\}$.

APPENDIX B

Proofs of Chapter 3

The procedure in Section 3.2.1 yields a confidence interval $[\hat{L}, \hat{U}]$ where $\hat{L}$ is defined by Equation (3.4) and $\hat{U}$ is defined by Equation (3.5). This section presents a proof that $[\hat{L}, \hat{U}]$ has asymptotic coverage at least $1 - \alpha$ as the number of scenarios $k \to \infty$, if the payoffs are normally distributed. In reality and in the example on which our experiments are run, payoffs are not normally distributed. However, sample averages of payoffs are approximately normal because of the central limit theorem if the sample sizes n_0 and N_i are large enough.

Theorem 1. *Suppose that for any scenario Z, the conditional distribution of the payoff X given Z is normal. Let $[\hat{L}, \hat{U}]$ represent the CI written in Equations (3.4) and (3.5), as produced by the procedure. Then* $\lim_{k\to\infty} \Pr\{\hat{L} \leq ES_{1-p} \leq \hat{U}\} \geq 1 - \alpha$.

The proof is within the framework of Chapter 2.2 for showing the asymptotic validity of a confidence interval generated by two-level simulation. Baysal and Staum (2008) show that the asymptotic probability that ES_{1-p} is contained in the outer-level confidence interval of Equation (2.3) is at least $1 - \alpha_o$ as $k \to \infty$. By the results of Chapter 2.2, it then suffices to construct a confidence region $\mathcal{V}$ such that

(1) the probability that $\mathcal{V}$ contains the vector of true values $\mathbf{V} = (V_1, V_2, \ldots, V_k)$ is at least $1 - \alpha_i$, and

(2) the two-level simulation confidence limits in Equations (3.4) and (3.5) arise as follows:

$$\hat{L} = \min_{\mathbf{v}\in\mathcal{V},\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' v_{\pi_v(i)} \quad \text{and} \quad \hat{U} = \max_{\mathbf{v}\in\mathcal{V},\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' v_{\pi_v(i)}, \tag{B.1}$$

where $\mathcal{S}(k)$ is defined in Equation (2.1) and π_v is a permutation of $\{1, 2, \ldots, k\}$ such that $v_{\pi_v(i)}$ is nondecreasing in i.

We first describe $\mathcal{V}$. Define the t-quantiles

$$z_{\text{lo}}(\ell) := t_{1-\alpha_{\text{lo}},\min_{i=\pi_0(1),\ldots,\pi_0(\ell)}\{N_i-1\}} \quad \text{and} \quad z_{\text{hi}} := t_{1-\alpha_{\text{hi}},\min_{i=\pi_1(1),\ldots,\pi_1(|I|)}\{N_i-1\}}.$$

The degrees of freedom in these formulae are the minimum degrees of freedom available in estimating any of the relevant standard deviations. Also define

$$B_0(\ell) := \max_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sqrt{\sum_{i=1}^{\ell} (w_i')^2 \frac{S^2_{\pi_0(i)}(N_{\pi_0(i)})}{N_{\pi_0(i)}}} \quad \text{and}$$

$$B_S(\ell) := \max_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sqrt{\sum_{i=1}^{\ell} (w_i')^2 \frac{S^2_{\pi_s(i)}(N_{\pi_s(i)})}{N_{\pi_s(i)}}}$$

to be used as bounds on standard deviations of weighted averages. Our confidence region $\mathcal{V}$ for $\mathbf{V}$ is the set containing all vectors $\mathbf{v}$ such that

$$\forall i \notin I,\ v_i \geq v_{\pi_v(\lceil kp \rceil)}, \tag{B.2}$$

$$\min_{\ell=\lfloor kp \rfloor,\ldots,\ell_{\max}} \left(\min_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\text{lo}}(\ell) B_0(\ell) \right) \leq \min_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' v_{\pi_v(i)}, \tag{B.3}$$

and

$$\max_{\ell=\ell_{\min},\ldots,\lceil kp\rceil}\left(\max_{\mathbf{w}\in\mathcal{S}_\ell(k)}\sum_{i=1}^{\ell}w_i'\bar{X}_{\pi_1(i)}(N_{\pi_0(i)})+z_{\mathrm{hi}}B_S(\ell)\right)\geq\max_{\mathbf{w}\in\mathcal{S}(k)}\sum_{i=1}^{k}w_i'v_{\pi_v(i)}. \tag{B.4}$$

This construction makes Equation (B.1) hold: compare Equation (3.4) with Equation (B.3) and Equation (3.5) with Equation (B.4).

The event that the confidence region $\mathcal{V}$ includes the true values $\mathbf{V}$ can be understood by plugging in $\mathbf{V}$ for $\mathbf{v}$ in the definition of $\mathcal{V}$. If we do so, then the constraints defining $\mathcal{V}$ take on the following interpretations.

- Equation (B.2) is equivalent to correct screening: $\gamma := \{\pi_V(1), \pi_V(2), \ldots, \pi_V(\lceil kp\rceil)\} \subseteq I$.
- Equation (B.3) implies that the two-level lower confidence limit $\hat{L} \leq \min_{\mathbf{w}\in\mathcal{S}(k)}\sum_{i=1}^{k} w_i'V_{\pi_V(i)}$, the outer-level lower confidence limit.
- Equation (B.4) implies that the two-level upper confidence limit $\hat{U} \geq \max_{\mathbf{w}\in\mathcal{S}(k)}\sum_{i=1}^{k} w_i'V_{\pi_V(i)}$, the outer-level upper confidence limit.

By the Bonferroni inequality, the probability that $\mathbf{V} \notin \mathcal{V}$ is bounded above by the sum of the probabilities

- that $\mathbf{V}$ does not satisfy Equation (B.2),
- that $\mathbf{V}$ does not satisfy Equation (B.3), and
- that $\mathbf{V}$ satisfies Equation (B.2) and does not satisfy Equation (B.4).

In Appendix B.1 we will show that the probability that $\mathbf{V}$ does not satisfy Equation (B.2) is bounded above by α_s. In Appendix B.2 we will show that the other two probabilities

are bounded above by α_{lo} and α_{hi}, respectively. Because $\alpha_i = \alpha_s + \alpha_{\text{lo}} + \alpha_{\text{hi}}$, this proves that $\Pr\{\mathbf{V} \in \mathcal{V}\} \geq 1 - \alpha_i$.

B.1. Screening

Here we show that the probability of correct screening $\Pr\{\gamma \subseteq I\} \geq 1 - \alpha_s$, where $\gamma = \{\pi_V(1), \cdots, \pi_V(\lceil kp \rceil)\}$. Let

$$B_{ij} := \mathbf{1}\{\bar{X}_i(n_0) > \bar{X}_j(n_0) + dS_{ij}(n_0)/\sqrt{n_0}\}$$

be the indicator function which is 1 when scenario j beats scenario i. We have

$$\Pr\{\gamma \subseteq I\} \geq \Pr\{\forall i \in \gamma, j \notin \gamma,\ B_{ij} = 0\} \geq 1 - \sum_{i \in \gamma} \sum_{j \notin \gamma} \Pr\{B_{ij} = 1\}$$

by the Bonferroni inequality. For $i \in \gamma$ and $j \notin \gamma$, $V_i \leq V_j$. Therefore each

$$\Pr\{B_{ij} = 1\} = \Pr\left\{ \frac{\bar{X}_i(n_0) - \bar{X}_j(n_0)}{S_{ij}(n_0)/\sqrt{n_0}} > d \right\} \leq \frac{\alpha_s}{\lceil kp \rceil\,(k - \lceil kp \rceil)},$$

using $d = t_{n_0 - 1, 1 - \alpha_s/(k - \lceil kp \rceil)\lceil kp \rceil}$.

B.2. Confidence Region

In this section, we deal with the second-stage inner-level simulation, after screening and restarting have occurred. We can think of the first stage as randomly generating a simulation problem which the second stage solves. The first stage produces I and N_i for each $i \in I$. This is an experimental design for the second stage, specifying which scenarios to consider and how many payoffs to simulate from each of them.

To prove $\mathcal{V}$ is a confidence region for $\mathbf{V}$, we first prove two lemmas. The first lemma addresses the following issue. On the left side of Equation (B.3), the minimization is over $\ell \in \{\lfloor kp \rfloor, \ldots, \ell_{\max}\}$ and then over $\mathbf{w} \in \mathcal{S}_\ell(k)$. On the right side, the minimization is over $\mathbf{w} \in \mathcal{S}(k)$, which is equivalent to minimization over $\ell \in \{\ell_{\min}, \ldots, \ell_{\max}\}$ and then over $\mathbf{w} \in \mathcal{S}_\ell(k)$, according to the definition of $\mathcal{S}(k)$ in Equation (2.1). We formulated Equation (B.3) in this way so that the procedure can save time by minimizing over the smaller range $\ell \in \{\lfloor kp \rfloor, \ldots, \ell_{\max}\}$ instead of $\{\ell_{\min}, \ldots, \ell_{\max}\}$. For the same reason, in Equation (B.4) on the left side, we maximized over $\ell \in \{\ell_{\min}, \ldots, \lceil kp \rceil\}$ instead of over $\{\ell_{\min}, \ldots, \ell_{\max}\}$. This formulation of the confidence region complicates the proof of Theorem 2, in which we use Lemma 1 to show that we can minimize and maximize over these smaller ranges and still get the desired coverage for the confidence region.

Lemma 1. *For any k-vector v and $\ell \in \{1, 2, \ldots, \lfloor kp \rfloor - 1\}$, there exists $\ell' \geq \lfloor kp \rfloor$ such that*

$$\min_{\mathbf{w} \in \mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' v_{\pi_v(i)} \geq \min_{\mathbf{w} \in \mathcal{S}_{\ell'}(k)} \sum_{i=1}^{\ell'} w_i' v_{\pi_v(i)}.$$

Similarly, for any k-vector v and $\ell \in \{\lceil kp \rceil + 1, \lceil kp \rceil + 2, \ldots, k\}$, there exists $\ell' \leq \lceil kp \rceil$ such that

$$\max_{\mathbf{w} \in \mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' v_{\pi_v(i)} \leq \max_{\mathbf{w} \in \mathcal{S}_{\ell'}(k)} \sum_{i=1}^{\ell'} w_i' v_{\pi_v(i)}.$$

Proof: We prove the first part of the lemma in detail. For any $\ell \in \{1, 2, \ldots, \lfloor kp \rfloor - 1\}$, define

$$\underline{\mathbf{w}}(\ell) := \left(w_{\ell,1}, w_{\ell,2}, \ldots, w_{\ell,\ell}, \frac{1-p}{k-\ell}, \ldots, \frac{1-p}{k-\ell} \right)$$

where $w_{\ell,1}, w_{\ell,2}, \ldots, w_{\ell,\ell}$ are chosen so that

$$\sum_{i=1}^{\ell} w'_{\ell,i} v_{\pi_v(i)} = \min_{\mathbf{w} \in \mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w'_i v_{\pi_v(i)}.$$

By optimality of $\mathbf{w}$, $w_{\ell,1} \leq w_{\ell,2} \leq \cdots \leq w_{\ell,\ell}$. Because $\ell < kp$ implies $(1-p)/(k-\ell) < (1-p)/(k-kp) = 1/k$ and the weights sum to 1, one of the weights must exceed $(1-p)/(k-\ell)$. Combining the previous two statements, there exists an integer s between 0 and ℓ such that $w_{\ell,i} < (1-p)/(k-\ell)$ for all $i = 1, \ldots, s$ and $w_{\ell,i} \geq (1-p)/(k-\ell)$ for all $i = s+1, \ldots, \ell$.

First we consider the case in which the tail probability $p < 0.5$. The following construction shows there is an integer $\ell' \geq \lfloor kp \rfloor$ and a weight vector $\mathbf{w}(\ell') \in \mathcal{S}_{\ell'}(k)$, whose ith component is to be denoted $w_{\ell',i}$, such that $\sum_{i=1}^{\ell} w'_i v_{\pi_v(i)} \geq \sum_{i=1}^{\ell'} w'_{\ell',i} v_{\pi_v(i)}$.

(1) Choose ℓ' to be the largest integer such that $\sum_{i=1}^{s} w_{\ell,i} + (\ell'-s)(1-p)/(k-\ell) \leq p$. From the definition of s and ℓ', $p < 0.5$, and $\ell < \lfloor kp \rfloor$, which implies $(1-p)/(k-\ell) < 1/k$, it follows that $\ell' = \ell' - s + s \leq \ell' - s + (\sum_{i=1}^{s} w_{\ell,i})(k-\ell)/(1-p) \leq p(k-\ell)/(1-p) < k-\ell$ and $(\ell'+1)/k > \sum_{i=1}^{s} w_{\ell,i} + (\ell'+1-s)(1-p)/(k-l) > p$. Putting these together, $k - \ell > \ell' \geq \lfloor kp \rfloor > \ell$. Initialize $\mathbf{w}(\ell')$ to

$$\left(w_{\ell,1}, \ldots, w_{\ell,s}, \frac{1-p}{k-\ell}, \ldots, \frac{1-p}{k-\ell}, w_{\ell,s+1}, \ldots, w_{\ell,\ell} \right),$$

where $w_{\ell,s+1}$ is the $(k+s+1-l)$th component. That is, this weight vector is derived from $\underline{\mathbf{w}}$ by switching the weights of components $s+1, \ldots, \ell$ with the weights of components $k-\ell+s+1, \ldots, k$. This weight vector may not be in

$\mathcal{S}_{\ell'}(k)$ because $\sum_{i=1}^{s} w_{\ell,i}+(\ell'-s)(1-p)/(k-\ell)$ may be less than p. The following steps adjust $\mathbf{w}(\ell')$ to make $\sum_{i=1}^{\ell'} w_{\ell',i} = p$. Initialize $m \leftarrow 0$.

(2) Do while $\sum_{i=1}^{\ell'} w_{\ell',i} + (w_{\ell,\ell-m} - (1-p)/(k-\ell)) \leq p$:

  (a) Switch the weights in components $\ell' - m$ and $k - m$: let $w_{\ell',\ell'-m} \leftarrow w_{\ell,\ell-m}$ and $w_{\ell',k-m} \leftarrow (1-p)/(k-\ell)$.

  (b) Set $m \leftarrow m + 1$, end while.

(3) Because of the termination criterion of Step 2, at the present point in the construction, $\sum_{i=1}^{\ell'} w_{\ell',i}+(w_{\ell,\ell-m} - (1-p)/(k-\ell)) > p$. From this fact, $\sum_{i=1}^{\ell} w_{\ell,i} = p$, and $\ell' > \ell$, it follows that at the present point, $m < \ell - s$ and $\ell' - s - m \geq 2$. Then there exists $\Delta \geq 0$ satisfying $w_{\ell,\ell-m} - \Delta > (1-p)/(k-\ell)$ and $\sum_{i=1}^{\ell'} w_{\ell',i} + \Delta = p$. Add Δ to component $\ell' - m$ and subtract it from component $k - m$: set $w_{\ell',\ell'-m} \leftarrow (1-p)/(k-\ell) + \Delta$ and $w_{\ell',k-m} \leftarrow w_{\ell,\ell-m} - \Delta$.

In the end, this produces a weight vector $\mathbf{w}(\ell') \in \mathcal{S}_{\ell'}(k)$ such that $\sum_{i=1}^{\ell} w'_i v_{\pi_v(i)} \geq \sum_{i=1}^{\ell'} w'_{\ell',i} v_{\pi_v(i)}$.

In the case where $p > 0.5$, the proof is similar. In this case there is an integer s' such that $\sum_{i=1}^{s'} w_{\ell,i} + (1-p) \leq p$. Likewise, the proof of the second part of the lemma is similar, except that we will be choosing $\ell' < \ell$. □

The next lemma provides a tool like a t-test for weighted sums of independent normal random variables.

Lemma 2. *Suppose X_{ij} are independent for $i = 1, \ldots, \ell$ and $j = 1, \ldots, N_i$, normally distributed with mean V_i and variance σ_i^2. Let $\bar{X}_i$ and S_i^2 be respectively the sample mean and sample variance of $X_{i1}, \ldots, X_{iN_i}$. Suppose $\mathbf{w}$ is a nonnegative ℓ-vector whose elements*

sum to 1 and π *is a permutation of* $i = 1, 2, \ldots, \ell$. *Define* $A := \sum_{i=1}^{\ell} w_i(\bar{X}_{\pi(i)} - V_{\pi(i)})$ *and* $S^2 := \sum_{i=1}^{\ell} w_i^2 \left(S_{\pi(i)}^2/N_{\pi(i)}\right)$. *If* $0 < \epsilon < 0.5$ *then* $\Pr\left\{A \geq -t_{1-\epsilon,\min_{i=1,\ldots,\ell}\{N_i-1\}} S\right\} \geq 1 - \epsilon$.

Proof: Because of the independence, A is normal with mean 0 and variance $\sum_{i=1}^{\ell} w_i^2 \sigma_{\pi(i)}^2/N_{\pi(i)}$, and it is independent of $S_1^2/N_1, S_2^2/N_2, \ldots, S_\ell^2/N_\ell$, which are themselves mutually independent. We can write

$$S^2 = \sum_{i=1}^{\ell} w_i^2 \left(\frac{S_{\pi(i)}^2}{N_{\pi(i)}}\right) = \sigma_A^2 \sum_{i=1}^{\ell} \frac{w_i^2}{\sigma_A^2} \left(\frac{\sigma_{\pi(i)}^2}{N_{\pi(i)}}\right) \left(\frac{S_{\pi(i)}^2}{\sigma_{\pi(i)}^2}\right)$$

where $\sigma_A^2 := \sum_{i=1}^{l} w_i^2 \sigma_{\pi(i)}^2/N_{\pi(i)}$. Also define $\lambda_i := (w_i^2/\sigma_A^2)(\sigma_{\pi(i)}^2/N_{\pi(i)})$. Notice that $\lambda_1, \ldots, \lambda_\ell$ are nonnegative weights that sum to 1. The distribution of $(N_{\pi(i)} - 1)S_{\pi(i)}^2/\sigma_{\pi(i)}^2$ is chi-squared with $N_{\pi(i)} - 1$ degrees of freedom. By a property of the t distribution, $t_{1-\epsilon,\min_{i=1,\ldots,\ell}\{N_i-1\}}^2 \geq t_{1-\epsilon,N_i-1}^2$ for all $i = 1, 2, \ldots, \ell$. Because $\epsilon < 0.5$,

$$\begin{aligned}
\Pr\left\{A \geq -t_{1-\epsilon,\min_{i=1,\ldots,\ell}\{N_i-1\}} S\right\} &= \frac{1}{2} + \frac{1}{2}\Pr\left\{\frac{A^2}{\sigma_A^2} \geq t_{1-\epsilon,\min_{i=1,\ldots,\ell}\{N_i-1\}}^2 \sum_{i=1}^{\ell} \frac{w_i^2}{\sigma_A^2} \left(\frac{\sigma_{\pi(i)}^2}{N_{\pi(i)}}\right) \left(\frac{S_{\pi(i)}^2}{\sigma_{\pi(i)}^2}\right)\right\} \\
&\geq \frac{1}{2} + \frac{1}{2}\Pr\left\{\frac{A^2}{\sigma_A^2} \geq \sum_{i=1}^{\ell} t_{1-\epsilon,N_{\pi(i)}-1}^2 \lambda_i \left(\frac{S_{\pi(i)}^2}{\sigma_{\pi(i)}^2}\right)\right\}.
\end{aligned}$$

By Banerjee's Theorem Banerjee (1961),

$$\Pr\left\{A \geq -t_{1-\epsilon,\min_{i=1,\ldots,\ell}\{N_i-1\}} S\right\} \geq \frac{1}{2} + \frac{1}{2}(1 - 2\epsilon) = 1 - \epsilon.$$

□

Using this lemma, we can prove the main result, that $\mathcal{V}$ defined by Equations (B.2), (B.3) and (B.4) is a confidence region for $\mathbf{V}$ with confidence level $1 - \alpha_i$, where $\alpha_i = \alpha_s + \alpha_{\text{hi}} + \alpha_{\text{lo}}$.

Theorem 2. *Suppose that for any scenario Z, the conditional distribution of the payoff X given Z is normal. If the payoffs are simulated independently, then* $\Pr\{\mathbf{V} \in \mathcal{V}\} \geq 1-\alpha_i$.

Proof: To condense notation, define $\mathcal{V}(\text{CS})$ as the set containing $\mathbf{v}$ satisfying Equation (B.2), which relates to screening, $\mathcal{L}$ as the set containing $\mathbf{v}$ satisfying Equation (B.3), which relates to the lower confidence limit, and $\mathcal{U}$ as the set containing $\mathbf{v}$ satisfying Equation (B.4), which relates to the upper confidence limit. Then the confidence region $\mathcal{V} = \mathcal{V}(\text{CS}) \cap \mathcal{L} \cap \mathcal{U}$ and the probability that it contains the vector $\mathbf{V}$ of true values is

$$\Pr\{\mathbf{V} \in \mathcal{V}\} = \Pr\{\mathbf{V} \in \{\mathcal{V}(\text{CS}) \cap \mathcal{L} \cap \mathcal{U}\}\} \geq 1 - \Pr\{\mathbf{V} \notin \mathcal{L}\} - \Pr\{\mathbf{V} \notin \{\mathcal{U} \cap \mathcal{V}(\text{CS})\}\}.$$

Define

$$\underline{\ell} := \arg \min_{\ell=\lfloor kp \rfloor,\ldots,\ell_{\max}} \left(\min_{\mathbf{w} \in \mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' V_{\pi_V(i)} \right).$$

By Lemma 1, $\underline{\ell}$ is the number of scenarios that would lead to the lower confidence limit for ES if the true values $\mathbf{V}$ were known:

$$\min_{\mathbf{w} \in \mathcal{S}(k)} \sum_{i=1}^{k} w_i' V_{\pi_V(i)} = \min_{\mathbf{w} \in \mathcal{S}_{\underline{\ell}}(k)} \sum_{i=1}^{\underline{\ell}} w_i' V_{\pi_V(i)}. \tag{B.5}$$

Also define the weight vector $\underline{\mathbf{w}}$ that minimizes ES using the true values of the $\underline{\ell}$ scenarios with the lowest sample averages after the first stage,

$$\underline{\mathbf{w}} := \arg \min_{\mathbf{w} \in \mathcal{S}_{\underline{\ell}}(k)} \sum_{i=1}^{\underline{\ell}} w_i' V_{\pi_0(i)}.$$

Because $V_{\pi_V(1)}, V_{\pi_V(2)}, \ldots, V_{\pi_V(\underline{\ell})}$ are the lowest $\underline{\ell}$ values among $V_1, V_2, \ldots, V_k$, while the elements of $\mathbf{w}'$ are negative, using any weight vector $\mathbf{w}$ and switching to the wrong ordering

of the true values decreases the weighted average, so

$$\min_{\mathbf{w}\in\mathcal{S}_{\underline{\ell}}(k)} \sum_{i=1}^{\underline{\ell}} w_i' V_{\pi_V(i)} \geq \min_{\mathbf{w}\in\mathcal{S}_{\underline{\ell}}(k)} \sum_{i=1}^{\underline{\ell}} w_i' V_{\pi_0(i)}. \tag{B.6}$$

Let $\mathcal{F}_0$ represent the information produced by the first stage of the procedure. Using Equations (B.5)–(B.6), the error probability related to the lower confidence limit

$$\begin{aligned}
&\Pr\{\mathbf{V}\notin\mathcal{L}|\mathcal{F}_0\}\\
&= \Pr\left\{\min_{\mathbf{w}\in\mathcal{S}(k)}\sum_{i=1}^{k} w_i' V_{\pi_V(i)} < \min_{\ell=\lfloor kp\rfloor,\ldots,\ell_{\max}}\left(\min_{\mathbf{w}\in\mathcal{S}_\ell(k)}\sum_{i=1}^{\ell} w_i'\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\mathrm{lo}}(\ell)B_0(\ell)\right)\middle|\mathcal{F}_0\right\}\\
&= \Pr\left\{\min_{\mathbf{w}\in\mathcal{S}_{\underline{\ell}}(k)}\sum_{i=1}^{\underline{\ell}} w_i' V_{\pi_V(i)} < \min_{\ell=\lfloor kp\rfloor,\ldots,\ell_{\max}}\left(\min_{\mathbf{w}\in\mathcal{S}_\ell(k)}\sum_{i=1}^{\ell} w_i'\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\mathrm{lo}}(\ell)B_0(\ell)\right)\middle|\mathcal{F}_0\right\}\\
&\leq \Pr\left\{\min_{\mathbf{w}\in\mathcal{S}_{\underline{\ell}}(k)}\sum_{i=1}^{\underline{\ell}} w_i' V_{\pi_0(i)} < \min_{\ell=\lfloor kp\rfloor,\ldots,\ell_{\max}}\left(\min_{\mathbf{w}\in\mathcal{S}_\ell(k)}\sum_{i=1}^{\ell} w_i'\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\mathrm{lo}}(\ell)B_0(\ell)\right)\middle|\mathcal{F}_0\right\}\\
&\leq \Pr\left\{\sum_{i=1}^{\underline{\ell}} \underline{w}_i' V_{\pi_0(i)} < \sum_{i=1}^{\underline{\ell}} \underline{w}_i'\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - z_{\mathrm{lo}}(\underline{\ell})B_0(\underline{\ell})\middle|\mathcal{F}_0\right\}\\
&= \Pr\left\{\sum_{i=1}^{\underline{\ell}} \underline{w}_i'(\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - V_{\pi_0(i)}) > z_{\mathrm{lo}}(\underline{\ell})B_0(\underline{\ell})\middle|\mathcal{F}_0\right\}.
\end{aligned}$$

Because we restart after the first stage, the second-stage data that we use here are independent of first-stage data, and in particular of the first-stage ordering $\pi_0(\cdot)$. Therefore the conditional distribution of $\sum_{i=1}^{\underline{\ell}} \underline{w}_i'(\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - V_{\pi_0(i)})$ given $\mathcal{F}_0$ is normal with mean 0 and variance $\sum_{i=1}^{\underline{\ell}} (\underline{w}_i')^2\sigma^2_{\pi_0(i)}/N_{\pi_0(i)}$. By Lemma 2,

$$\Pr\left\{\sum_{i=1}^{\underline{\ell}} \underline{w}_i'(\bar{X}_{\pi_0(i)}(N_{\pi_0(i)}) - V_{\pi_0(i)}) > z_{\mathrm{lo}}(\underline{\ell})B_0(\underline{\ell})\middle|\mathcal{F}_0\right\} \leq \alpha_{\mathrm{lo}}.$$

Thus $\Pr\{\mathbf{V}\notin\mathcal{L}\} = \mathrm{E}\left[\Pr\{\mathbf{V}\notin\mathcal{L}|\mathcal{F}_0\}\right] \leq \alpha_{\mathrm{lo}}$.

Similarly, turning our attention from the lower confidence limit to the upper confidence limit and screening,

$$\begin{aligned}\Pr\{\mathbf{V} \notin \mathcal{U} \cap \mathcal{V}(\mathrm{CS})\} &= \Pr\{\mathbf{V} \notin \mathcal{V}(\mathrm{CS})\} + \Pr\{\mathbf{V} \in \mathcal{V}(\mathrm{CS}), \mathbf{V} \notin \mathcal{U}\} \\ &\leq \alpha_s + \mathrm{E}\left[\Pr\{\mathbf{V} \notin \mathcal{U} | \mathcal{F}_0\} \mathbf{1}\{\mathbf{V} \in \mathcal{V}(\mathrm{CS})\}\right].\end{aligned}$$

Define

$$\bar{\ell} := \arg \max_{\ell=\ell_{\min},\ldots,\lceil kp \rceil} \left\{ \max_{\mathbf{w} \in \mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' V_{\pi_V(i)} \right\} \quad \text{and} \quad \bar{\mathbf{w}} := \arg \max_{\mathbf{w} \in \mathcal{S}_{\bar{\ell}}(k)} \sum_{i=1}^{\bar{\ell}} w_i' V_{\pi_V(i)}.$$

By Lemma 1, the upper confidence limit for ES if the true values $\mathbf{V}$ were known is attained by using $\bar{\ell}$ scenarios and weight vector $\bar{\mathbf{w}}$:

$$\max_{\mathbf{w} \in \mathcal{S}(k)} \sum_{i=1}^{k} w_i' V_{\pi_V(i)} = \sum_{i=1}^{\bar{\ell}} \bar{w}_i' V_{\pi_V(i)}. \tag{B.7}$$

As shown by Baysal and Staum (2008), by optimality of $\bar{\mathbf{w}}$, $\bar{w}_1', \bar{w}_2', \ldots, \bar{w}_{\bar{\ell}}'$ is negative and non-decreasing, so

$$\sum_{i=1}^{\bar{\ell}} \bar{w}_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) \geq \sum_{i=1}^{\bar{\ell}} \bar{w}_i' \bar{X}_{\pi_V(i)}(N_{\pi_V(i)}). \tag{B.8}$$

Using Equations (B.7)–(B.8), the error probability related to the upper confidence limit

$$
\begin{aligned}
&\Pr\{\mathbf{V} \notin \mathcal{U} | \mathcal{F}_0\} \\
&= \Pr\left\{ \max_{\mathbf{w}\in\mathcal{S}(k)} \sum_{i=1}^{k} w_i' V_{\pi_V(i)} > \max_{\ell=\ell_{\min},\ldots,\lceil kp \rceil} \left\{ \max_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) + z_{\mathrm{hi}} B_S(\ell) \right\} \middle| \mathcal{F}_0 \right\} \\
&= \Pr\left\{ \sum_{i=1}^{\bar{\ell}} \bar{w}_i' V_{\pi_V(i)} > \max_{\ell=\ell_{\min},\ldots,\lceil kp \rceil} \left\{ \max_{\mathbf{w}\in\mathcal{S}_\ell(k)} \sum_{i=1}^{\ell} w_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) + z_{\mathrm{hi}} B_S(\ell) \right\} \middle| \mathcal{F}_0 \right\} \\
&\leq \Pr\left\{ \sum_{i=1}^{\bar{\ell}} \bar{w}_i' V_{\pi_V(i)} > \sum_{i=1}^{\bar{\ell}} \bar{w}_i' \bar{X}_{\pi_1(i)}(N_{\pi_1(i)}) + z_{\mathrm{hi}} B_S(\bar{\ell}) \middle| \mathcal{F}_0 \right\} \\
&\leq \Pr\left\{ \sum_{i=1}^{\bar{\ell}} \bar{w}_i' V_{\pi_V(i)} > \sum_{i=1}^{\bar{\ell}} \bar{w}_i' \bar{X}_{\pi_V(i)}(N_{\pi_V(i)}) + z_{\mathrm{hi}} B_S(\bar{\ell}) \middle| \mathcal{F}_0 \right\} \\
&= \Pr\left\{ \sum_{i=1}^{\bar{\ell}} \bar{w}_i' (\bar{X}_{\pi_V(i)}(N_{\pi_V(i)}) - V_{\pi_V(i)}) < -z_{\mathrm{hi}} B_S(\bar{\ell}) \middle| \mathcal{F}_0 \right\}.
\end{aligned}
$$

The conditional distribution of $\sum_{i=1}^{\bar{\ell}} \bar{w}_i' (\bar{X}_{\pi_V(i)}(N_{\pi_V(i)}) - V_{\pi_V(i)})$ given $\mathcal{F}_0$ is normal with mean 0 and variance $\sum_{i=1}^{\bar{\ell}} (\bar{w}_i')^2 \sigma^2_{\pi_V(i)}/N_{\pi_V(i)}$. If $\mathbf{V} \in \mathcal{V}(\mathrm{CS})$, then

$$
B_S^2(\bar{\ell}) = \left(\bar{s}\Delta(\bar{\ell})\right)^2 = \left(\max_{i=1,\ldots,k} \frac{S_i^2(N_i)}{N_i} \right) \left(\max_{\mathbf{w}\in\mathcal{S}(\bar{\ell})} \sum_{i=1}^{\bar{\ell}} (w_i')^2 \right) \geq \sum_{i=1}^{\bar{\ell}} \frac{(w_i')^2 S^2_{\pi_V(i)}}{N_{\pi_V(i)}},
$$

and so by Lemma 2,

$$
\Pr\left\{ \sum_{i=1}^{\bar{\ell}} \bar{w}_i' (\bar{X}_{\pi_V(i)}(N_{\pi_V(i)}) - V_{\pi_V(i)}) < -z_{\mathrm{hi}} B_S(\bar{\ell}) \middle| \mathcal{F}_0 \right\} \mathbf{1}\{\mathbf{V} \in \mathcal{V}(\mathrm{CS})\} \leq \alpha_{\mathrm{hi}}.
$$

Therefore $\mathrm{E}\left[\Pr\{\mathbf{V} \notin \mathcal{U} | \mathcal{F}_0\} \mathbf{1}\{\mathbf{V} \in \mathcal{V}(\mathrm{CS})\}\right] \leq \alpha_{\mathrm{hi}}$, and so $\Pr\{\mathbf{V} \notin \mathcal{U} \cap \mathcal{V}(\mathrm{CS})\} \leq \alpha_s + \alpha_{\mathrm{hi}}$.

In total, $\Pr\{\mathbf{V} \in \mathcal{V}(\mathrm{CS}) \cap \mathcal{L} \cap \mathcal{U}\} \geq 1 - \alpha_{\mathrm{lo}} - \alpha_s - \alpha_{\mathrm{hi}} = 1 - \alpha_i$. □

APPENDIX C

Appendix of Chapter 4

C.1. Proof of Equation (4.9)

The number of comparisons needed to be done, M, can be predicted as:

$$\mathrm{E}[M] = \mathrm{E}\left[\sum_{i'=\lceil kp\rceil+1}^{k} M_{\pi_0(i')}\right].$$

As shown in Figure C.1, by assuming the probability of the event that the market scenario $\pi_0(i')$ is beaten by $\pi_0(j')$ is given by the following step function

$$\Pr\{\pi_0(j') \text{ beats } \pi_0(i')\} = \begin{cases} 1, & j' \le b; \\ \bar{p}_B, & b < j' < i' - e - 1; \\ 0, & i' - e - 1 \le j' \le i' - 1. \end{cases} \tag{C.1}$$

we can simplify the calculation of $\mathrm{E}[M_{\pi_0(i')}]$ while keeping the approximation close to reality. In Equation (C.1), b and e depend on i'. For simplicity of notation, we do not explicitly show such dependence. Variables b and e can be computed through the procedure described in Section 4.2.2, as can the average probability $\bar{p}_B$.

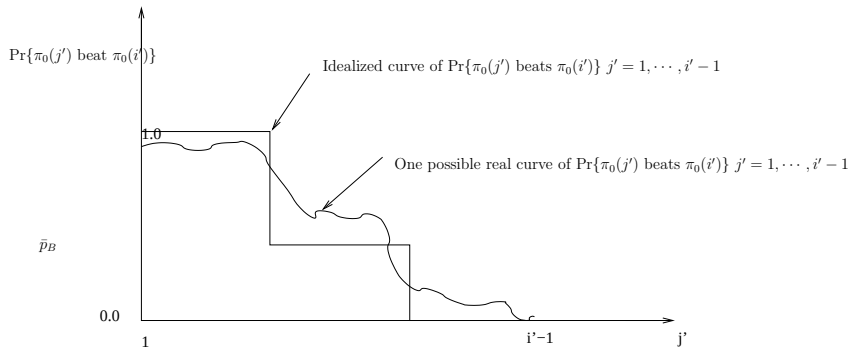

Figure C.1. Probability of $\pi_0(j')$ Beating $\pi_0(i')$

Under all the above assumptions, we have

(C.2)

$$\mathrm{E}[M_{\pi_0(i')}] = \begin{cases} \lceil kp \rceil, & \text{when } b > \lceil kp \rceil; \\ i' - 1, & \text{when } i' - e - 1 < \lceil kp \rceil; \\ b + \sum_{h=\lceil kp \rceil - b}^{i'-1-e-b} h \binom{h-1}{\lceil kp \rceil - b - 1} \bar{p}_B^{\lceil kp \rceil - b} (1-\bar{p}_B)^{h+b-\lceil kp \rceil} & \\ \quad + (1 - \mathrm{NB}_{\bar{p}_B}(\lceil kp \rceil - b, i' - e - \lceil kp \rceil - 1))(i' - b - 1). & \text{o.w.} \end{cases}$$

Since $h\binom{h-1}{\lceil kp \rceil - b - 1} \bar{p}_B^{\lceil kp \rceil - b} (1-\bar{p}_B)^{h+b-\lceil kp \rceil} = \frac{\lceil kp \rceil - b}{\bar{p}_B} \binom{h}{\lceil kp \rceil - b} \bar{p}_B^{\lceil kp \rceil - b + 1} (1-\bar{p}_B)^{h+b-\lceil kp \rceil}$, we can further simplify Equation (C.2) to Equation (4.9).

C.2. Parameters k_0 and n_{00} of the Test Run

Until now, we have assumed that $\mathcal{T}(k_0, n_{00})$ contains information necessary for the prediction of k and n_0. Here we discuss how to determine the k_0 and n_{00} of the test run upon which k and n_0 are predicted.

For k_0, we have demonstrated that when the number of scenarios $k \geq 40/p$, the coverage rate of the estimated confidence interval of the expected shortfall ES_{1-p} is no

less than the nominal confidence level (Lan et al., 2007a). That is, when $k \geq 40/p$, the asymptotic correctness of our procedure holds. We then also require that $k_0 \geq 40/p$. In our experiments, $p = 0.01$, we took $k_0 = 4000$, which worked well for all tested problems.

The behavior of screening procedure under (k, n_0) is predicted by using $\frac{\bar{X}_i(n_{00})-\bar{X}_j(n_{00})}{S_{ij}(n_{00})}$, $i, j = 1, \ldots, k_0$ repeatedly. We need to set n_{00} large enough such that $\frac{\bar{X}_i(n_{00})-\bar{X}_j(n_{00})}{S_{ij}(n_{00})}$ is close to its true value. Suppose that $\frac{\tau_{ij}^2}{n_{00}}$ is the variance of $\frac{\bar{X}_i(n_{00})-\bar{X}_j(n_{00})}{S_{ij}(n_{00})}$, then we can determine n_{00} by

$$n_{00} \geq \frac{\tau_{ij}^2}{\frac{(\bar{X}_i(n'_{00})-\bar{X}_j(n'_{00}))^2}{S_{ij}^2(n'_{00})}a^2} \ \forall i, j = 1, \ldots, k_0,$$

where a stands for the ratio of standard variance to mean value and is set to 0.05 in all experiments, n'_{00} is the size of samples used to estimate n_{00}, usually chosen as 160 or even larger because all the data can be reused in the test run. Let $Y_h(i, j) = X_{i,h} - X_{j,h}$ $h = 1, \ldots, n'_{00}$. We often drop (i, j) and write $Y_h(i, j)$ as Y_h when the context is clear. Suppose $y_1, y_2, \ldots$ are the non-central moments of Y_h. We denote $\hat{y}_1, \hat{y}_2, \ldots$ as the ordinary estimators of $y_1, y_2, \ldots$, that is, $\hat{y}_1 = \sum_{h=1}^{n'_{00}} Y_h$, $\hat{y}_2 = \sum_{h=1}^{n'_{00}} Y_h^2$ and so on. From the Δ-method, we can estimate τ_{ij}^2 by

$$\hat{\tau}_{ij}^2 = \frac{1}{(\hat{y}_2 - \hat{y}_1^2)^3} \left\{ \hat{y}_1^2\hat{y}_4 - \frac{1}{4}\hat{y}_1^2\hat{y}_2^2 - \hat{y}_1\hat{y}_2\hat{y}_3 + \hat{y}_2^3 \right\}.$$

Then we can estimate the lower bound of n_{00} by

$$\max_{i,j} \frac{400}{(\hat{y}_2 - \hat{y}_1^2)^2\hat{y}_1^2} \left\{ \hat{y}_1^2\hat{y}_4 - \frac{1}{4}\hat{y}_1^2\hat{y}_2^2 - \hat{y}_1\hat{y}_2\hat{y}_3 + \hat{y}_2^3 \right\}. \tag{C.3}$$

Usually $i = 1, j \in \{\pi_0(k_0 p), \pi_0(k_0)\}$ is taken instead of the whole possible set of i, j to save computation. See Appendix C.2.1 for the procedure on how to determine n_{00} adaptively and generate the test run data $\mathcal{T}(k_0, n_{00})$.

C.2.1. Procedure for $\mathcal{T}(k_0, n_{00})$

We introduce the procedure used to generate the test run data $\mathcal{T}(k_0, n_{00})$ adaptively.

(1) **Initialization**:

Set $n_{00} \leftarrow 240$ (default value), $k_0 \leftarrow 40/p$ and $n_{\max} = 10^9/k_0$ (limited by the memory the computer can provide).

Set $a \leftarrow 0$.

(2) **Generating Scenarios**:

Generate k_0 scenarios independently and calculate the unit time used on generating market scenario R_{sce}.

(3) **Simulation of Portfolio Payoffs**

(a) Set $n_{\text{old}} \leftarrow 0$

(b) Do:

(i) Generate additional $n_{00} - n_{\text{old}}$ payoffs for each market scenario with Common Random Numbers.

(ii) Set $n_{\text{old}} \leftarrow n_{00}$.

(iii) Update n_{00} according to Equation (C.3).

(iv) Set $n_{00} \leftarrow n_{00} \wedge n_{\max}$ and $a \leftarrow a + 1$.

While($n_{00} > n_{\text{old}}$ and $a < 3$)

(c) Set $n_{00} \leftarrow n_{\text{old}}$.

(d) Calculate the unit time spent on simulation of one payoff R_{sam}.

(4) **Screening**:

Perform the screening operation on all the k_0 scenarios.

Record $\bar{X}_i(n_{00}), S^2(n_{00}), \pi_0(\cdot)$ and $S^2_{ij}(n_{00})$.

Calculate the time related coefficients $R_{\text{scr1}}, R_{\text{scrsort}}$ and R_{scr2}.

(5) **Computing the Empirical Likelihood Estimator**:

Compute the Empirical Likelihood estimation with data $\bar{X}_{\pi_0(1)}(n_{00}), \ldots, \bar{X}_{\pi_0(|I|)}(n_{00})$.

Calculate coefficient E_{out} and R_{com}.

C.3. Experimental Data

C.3.1. Experiments for Given C Procedures

Table C.1: Experiment Results for Problem 1 with $C = 16000000$

Width of CI when C=16000000			
k	n_0	Average CI Width	Variance of CI Width
1000	80	0.73	4.9e-02
32000	70	0.155	5.7e-05
128000	100	0.119	2.0e-06
32000	80	0.156	5.7e-05
128000	30	0.68	2.4e-01
4000	100	0.41	3.7e-03
128000	50	0.094	1.4e-06
4000	150	0.41	3.7e-03
128000	60	0.097	2.6e-06

Continued on next page

Table C.1 – continued ($C = 16000000$)

k	n_0	Average CI Width	Variance of CI Width
4000	300	0.41	3.7e-03
128000	70	0.101	2.0e-06
4000	30	0.74	2.8e-01
128000	72	0.101	2.7e-06
4000	500	0.41	3.7e-03
128000	74	0.102	1.9e-06
4000	50	0.41	3.7e-03
128000	76	0.103	1.7e-06
4000	60	0.41	3.7e-03
128000	80	0.105	3.0e-06
4000	70	0.41	3.7e-03
16000	100	0.211	2.4e-04
4000	800	0.41	3.7e-03
16000	10	0.41	3.4e-03
4000	80	0.41	3.7e-03
16000	150	0.21	2.4e-04
125262	67	0.099	3.7e-06
16000	20	0.46	2.0e-02
16000	300	0.21	2.4e-04
16000	30	0.47	6.9e-02
16000	500	0.22	2.3e-04
500	80	0.93	1.7e-01
16000	50	0.21	2.4e-04
64000	100	0.120	9.4e-06
16000	60	0.21	2.4e-04

Continued on next page

Table C.1 – continued ($C = 16000000$)

k	n_0	Average CI Width	Variance of CI Width
64000	150	0.126	1.2e-05
16000	70	0.21	2.4e-04
64000	30	0.37	5.1e-02
16000	800	0.23	2.6e-04
64000	50	0.116	1.3e-05
16000	80	0.21	2.4e-04
64000	60	0.117	1.2e-05
171803	74	0.112	3.4e-06
64000	70	0.117	1.3e-05
2000	80	0.56	1.3e-02
64000	80	0.118	1.2e-05
250	80	0.78	1.7e-01
8000	100	0.29	9.3e-04
256000	30	1.5	1.5e+00
8000	150	0.29	9.3e-04
256000	50	0.106	3.1e-06
8000	300	0.29	9.4e-04
256000	60	0.204	1.0e-05
8000	30	0.60	1.4e-01
300	80	0.83	2.0e-01
8000	500	0.30	9.4e-04
32000	100	0.156	5.6e-05
8000	50	0.29	9.3e-04
32000	150	0.157	5.5e-05
8000	60	0.29	9.3e-04

Continued on next page

Table C.1 – continued ($C = 16000000$)

k	n_0	Average CI Width	Variance of CI Width
32000	300	0.165	5.4e-05
8000	70	0.29	9.3e-04
32000	30	0.41	3.1e-02
8000	800	0.30	9.4e-04
32000	50	0.155	5.7e-05
8000	80	0.29	9.3e-04
32001	60	0.155	5.7-05

C.3.1.1. Problem 1.

Table C.2: Experiment Results for Problem 2 using Strategy CRN2 with $C = 128000000$

Width of CI when C=128000000			
k	n_0	Average CI Width	Variance of CI Width
9140	8031	4.6	1.5e+00
4000	8000	5.5	2.3e+00
12000	8000	6.7	2.1e+01
9930	7014	6.2	1.5e+01
14000	8000	9.6	6.7e+01
6000	12000	4.7	9.2e-01
16000	4000	16	9.0e+01
64000	40	69	2.7e+02
16000	40	37	2.8e+02
8000	12000	5.0	7.7e-01
32000	40	49	2.7e+02
8000	4000	9.7	2.0e+01
4000	12000	5.1	8.4e-01
8000	8000	5.4	6.1e+00
4000	4000	8	1.0e+01
6872	9408	5.0	2.7e+00
4000	40	28	2.9e+02
9000	12000	5.5	2.0e+00

C.3.1.2. Problem 2.

Table C.3: Experiment Results for Problem 2 using Strategy CRN with $C = 128000000$

Width of CI when C=128000000			
k	n_0	Average CI Width	Variance of CI Width
14000	8000	6.7	2.7e-01
4000	40	28	2.2e+02
16000	2000	9	2.5e+01
4000	8000	5.2	7.9e-01
16000	4000	5.3	3.4e+00
6000	12000	4.8	3.9e-01
16000	40	36	2.2e+02
64000	40	66	2.2e+02
16000	6000	5.6	6.7e-01
8000	2000	7.1	8.5e+00
24000	2000	12	5.2e+01
8000	4000	4.8	7.1e-01
24000	4000	8.5	1.6e+01
8000	40	31	2.2e+02
32000	2000	17	1.1e+02
8000	6000	4.4	2.2e-01
32000	40	47	2.1e+02
8000	8000	4.5	2.5e-01

Table C.4: Experiment Results for Problem 2 using Strategy NAIVE with $C = 128000000$

Width of CI when C=128000000			
k	n_0	Average CI Width	Variance of CI Width
12000	8000	9.2	6.7e+00
5133	13713	5.8	3.7e-01
14000	8000	13.8	2.5e+01
7625	8633	6.1	7.4e-01
16000	4000	26	3.7e+02
6000	12000	5.6	2.4e-01
16000	40	62	4.0e+02
6000	14000	5.9	2.5e-01
32000	4000	0.0	0.0e+00
6000	18000	7.3	1.9e-01
32000	40	80	4.2e+02
64000	40	110	4.2e+02
4000	12000	5.7	8.3e-01
8000	12000	6.6	2.2e-01
4000	14000	5.8	8.4e-01
8000	4000	13.9	8.1e+01
4000	18000	5.926545	8.1e-01
8000	40	51	4.0e+02
4000	4000	11	3.5e+01
8000	8000	6.5	1.6e+00
4000	40	46	3.9e+02
4642	14572	5.8	5.7e-01

Continued on next page

Table C.4 – continued ($C = 128000000$)

k	n_0	Average CI Width	Variance of CI Width
4000	8000	6.1	1.1e+00
9000	12000	8.0	4.4e-01

Table C.5: Experiment Results for Problem 2 using Strategy NATURE with $C = 128000000$

Width of CI when C=128000000			
k	n_0	Average CI Width	Variance of CI Width
14000	8000	4.9	1.0e-01
32000	40	36	2.0e+02
16000	2000	4.9	3.5e+00
32000	6000	0.0	0.0e+00
16000	4000	3.7	2.1e-01
4000	40	26	2.9e+02
16000	40	30	3.0e+02
4000	8000	4.9	7.3e-01
16000	6000	4.2	9.2e-02
15525	4943	3.7	1.0e-01
18000	5000	4.1	9.2e-02
6000	12000	4.4	3.6e-01
20000	4000	3.9	1.5e-01
64000	40	48	2.9e+02
24000	2000	6	8.7e+00
8000	2000	4.7	8.8e-01
24000	4000	4.7	3.3e-01
8000	4000	4.1	2.6e-01
24000	6000	0.0	0.0e+00
8000	40	27	2.9e+02
32000	2000	7.8	2.1e+01
8000	6000	4.0	1.8e-01

Continued on next page

Table C.5 – continued ($C = 128000000$)

k	n_0	Average CI Width	Variance of CI Width
32000	4000	0.0	0.0e+00
8000	8000	4.0	2.0e-01

C.3.2. Experiments for Given T Procedures

Table C.6: Experiment Results for Problem 1 with $T = 60$

Width of CI when T=60				
k	n_0	Width	Var	Accomplishment Rate
1000	80	0.70	4.8e-02	100.0%
4000	80	0.39	3.7e-03	100.0%
112000	120	0.0	0.0e+00	0.0%
93820	68	0.091	6.1e-06	100.0%
112000	60	0.10	0.0e+00	5.0%
48000	120	0.119	2.7e-05	100.0%
144000	120	0.0	0.0e+00	0.0%
48000	40	0.15	9.8e-03	100.0%
144000	40	0.0	0.0e+00	0.0%
48000	60	0.118	2.7e-05	100.0%
144000	60	0.0	0.0e+00	0.0%
48000	80	0.118	2.5e-05	100.0%
144000	80	0.0	0.0e+00	0.0%
500	80	0.91	1.7e-01	100.0%
16000	120	0.20	2.9e-04	100.0%
8000	120	0.28	1.0-03	100.0%
16000	40	0.20	2.9e-04	100.0%
8000	40	0.28	1.0e-03	100.0%
16000	60	0.20	2.9e-04	100.0%
8000	60	0.28	1.0e-03	100.0%
16000	80	0.20	2.4e-04	100.0%

Continued on next page

Table C.6 – continued ($T = 60$)

k	n_0	Width	Var	Accomplishment Rate
8000	80	0.28	9.3e-04	100.0%
2000	80	0.55	1.3e-02	100.0%
96000	120	0.0	0.0e+00	0.0%
24000	120	0.165	8.4e-05	100.0%
96000	40	0.13	9.0e-03	80.0%
24000	40	0.19	9.8e-03	100.0%
96000	60	0.089	5.3e-06	100.0%
24000	60	0.165	8.5e-05	100.0%
96000	80	0.110	5.2e-05	100.0%
24000	80	0.165	8.3e-05	100.0%

C.3.2.1. Problem 1.

Table C.7: Experiment Results for Problem 2 using Strategy CRN2 with $T = 240$

Width of CI when T=240				
k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
1000	12000	8.7	4.1e+00	100.0%
10515	5436	4.0	2.4e-01	95.0%
1000	6000	8.8	3.6e+00	100.0%
4813	11233	4.6	6.0e-01	100.0%
1000	8000	8.7	4.0e+00	100.0%
6000	1000	13	3.9e+01	100.0%
11233	4813	4.1	2.9e-01	95.0%
6000	12000	4.4	3.4e-01	100.0%
12000	1000	0.0	0.0e+00	0.0%
6000	2000	7.5	2.2e+01	100.0%
12000	2000	6.2	2.4e+01	65.0%
6000	4000	4.6	6.3e-01	100.0%
12000	4000	4.4	2.5e+00	100.0%
6000	6000	4.4	3.8e-01	100.0%
12000	6000	4.8	5.1e+00	100.0%
6000	8000	4.4	4.2e-01	100.0%
12000	8000	5.6	3.0e+00	100.0%
8000	1000	20	2.8e+02	70.0%
2000	12000	6.7	2.0e+00	100.0%
8000	12000	4.3	2.2e-01	100.0%
2000	6000	6.8	2.0e+00	100.0%
8000	2000	11	1.7e+02	100.0%

Continued on next page

Table C.7 – continued ($T = 240$)

k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
2000	8000	6.8	2.0e+00	100.0%
8000	4000	4.7	1.8e+00	100.0%
4000	12000	4.9	7.8e-01	100.0%
8000	6000	4.1	2.0e-01	100.0%
4000	6000	5.0	8.3e-01	100.0%
8000	8000	4.1	2.3e-01	100.0%
4000	8000	4.9	8.2e-01	100.0%

C.3.2.2. Problem 2.

Table C.8: Experiment Results for Problem 2 using Strategy CRN with $T = 240$

Width of CI when T=240				
k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
12000	8000	5.822798	1.6e+01	40.0%
500	8000	11	3.1e+01	100.0%
2000	8000	6.6	6.8e-01	100.0%
6000	8000	4.6	1.4e+00	100.0%
4000	8000	5.1	5.8e-01	100.0%
8000	8000	3.6	2.3e+00	90.0%
6802	8305	4.8	3.0e+00	95.0%

Table C.9: Experiment Results for Problem 2 using Strategy CRN with $T = 240$

Width of CI when T=240				
k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
11205	6723	4.0	5.4e-01	100%
10227	6706	4.0	8.5e-02	100%
9628	8353	3.9	1.1e-01	100%
9553	7865	4.2	1.0e-01	100%
8536	8550	4.0	1.9e-01	100%
10746	6218	4.0	1.3e-01	100%
10030	7436	3.9	1.0e-01	100%
8564	9277	4.0	7.5e-02	100%
10824	6651	3.9	8.1e-02	100%
11425	5844	3.9	8.2e-02	100%
9195	7857	3.9	2.2e-01	100%
11107	6547	4.0	1.7e-01	100%
11598	5774	3.9	2.1e-01	100%
10124	7284	3.9	1.4e-01	100%
9976	6875	3.9	1.0e-01	100%
9297	7750	4.0	1.3e-01	100%
9123	7872	3.8	8.8e-02	100%
10895	6788	3.9	8.5e-02	100%
9177	8328	4.009364	1.0e-01	100%
10747	6625	3.9	1.7e-01	100%

Table C.10: Experiment Results for Problem 2 using Strategy CRN with $T = 120$

Width of CI when T=120				
k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
1000	3000	9.3	6.6e+00	100.0%
4000	8000	5.8	6.2e-01	100.0%
1000	8000	9.6	6.6e+00	100.0%
5350	7461	9	6.6e+01	90.0%
2000	3000	7.2	2.0e+00	100.0%
500	3000	11	1.8e+01	100.0%
2000	8000	7.0	5.8e-01	100.0%
500	8000	11	3.1e+01	100.0%
250	3000	10	2.6e+01	100.0%
6000	3000	5.3	2.8e+00	100.0%
250	8000	11	9.6e+01	100.0%
6000	8000	4.9	3.6e+00	90.0%
4000	3000	5.7	8.0e-01	100.0%

Table C.11: Experiment Results for Problem 2 using Strategy NAIVE with $T = 240$

Width of CI when T=240				
k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
10000	6000	18	1.2e+02	30.0%
6000	8000	10	2.0e+01	95.0%
10000	8000	0.0	0.0e+00	0.0%
7248	5734	10	2.0e+01	55.0%
2000	6000	8.9	3.9e+00	100.0%
8000	1000	24	1.1e+03	2.0%
4000	12000	7.0	7.9e-01	100.0%
8000	12000	0.0	0.0e+00	0.0%
4000	6000	9	2.3e+01	100.0%
8000	2000	10	8.9e+01	3.0%
4000	8000	7.2	1.9e+00	100.0%
8000	500	84	2.1e+02	100.0%
5057	9035	7.5	5.9e+00	100.0%
8000	6000	15	9.7e+01	59.0%
6000	12000	57	9.2e+02	55.0%
8000	8000	25	3.8e+02	46.0%
6000	6000	20	1.3e+03	80.0%

Table C.12: Experiment Results for Problem 2 using Strategy NAIVE with $T = 240$

Width of CI when T=240				
k	n_0	Average CI Width	Variance of CI Width	Accomplishment Rate
4000	14071	6.9	6.6e-01	100%
4000	14092	7.0	7.7e-01	100%
4000	12603	6.7	4.2e-01	100%
4000	17172	8.1	1.1e+00	100%
5080	8701	6.8	1.4e+00	100%
4980	10183	7.1	6.7e-01	100%
4504	10680	7.6	4.1e+00	100%
5595	8383	7.0	8.7e-01	95%
5584	8400	8.6	1.9e+01	100%
4518	10267	6.6	5.7e-01	100%
4538	10888	7.5	3.3e+00	100%
4641	9728	6.6	8.3e-01	100%
4000	13639	6.9	6.1e-01	100%
4000	13155	6.9	5.1e-01	100%
4681	10269	6.6	5.1e-01	100%
5593	8325	7.0	7.7e-01	95%
4481	11187	6.8	6.3e-01	100%
5423	9236	7.1	8.5e-01	95%
5261	9272	7.9	7.3e+00	100%
4047	14343	7.3	6.5e-01	100%

Table C.13: Experiment Results for Problem 2 using Strategy NATURE with $T = 240$

Width of CI when T=240				
k	n_0	Width	Var	Accomplished Rate
1000	12000	8.7	3.8e+00	100.0%
4000	8000	4.9	7.4e-01	100.0%
1000	6000	8.7	3.6e+00	100.0%
4813	11233	4.7	6.0e-01	100.0%
1000	8000	8.7	3.7e+00	100.0%
6000	1000	14	2.6e+01	100.0%
11233	4813	4.1	1.3e-01	100.0%
6000	12000	4.6	3.4e-01	100.0%
12000	1000	2.5	1.3e+01	10.0%
6000	2000	5.1	1.3e+00	100.0%
12000	2000	4.5	7.6e-01	85.0%
6000	4000	4.4	4.2e-01	100.0%
12000	4000	4.2	7.4e-01	100.0%
6000	6000	4.3	4.0e-01	100.0%
12000	6000	5.2	1.5e-01	100.0%
6000	8000	4.4	4.1e-01	100.0%
12000	8000	0.0	0.0e+00	0.0%
8000	1000	24	1.1e+03	45.0%
2000	12000	6.7	2.2e+00	100.0%
8000	12000	0.0	0.0e+00	0.0%
2000	6000	6.8	2.1e+00	100.0%
8000	2000	8	1.2e+02	100.0%

Continued on next page

Table C.13 – continued ($T = 240$)

k	n_0	Width	Var	Accomplished Rate
2000	8000	6.7	2.1e+00	100.0%
8000	4000	4.1	2.7e-01	100.0%
4000	12000	4.9	7.3e-01	100.0%
8000	6000	4.1	1.7e-01	100.0%
4000	6000	4.9	7.4e-01	100.0%
8000	8000	4.3	1.8e-01	100.0%