

# Utility Analysis of Parallel Simulation

David M. Nicol  
Dartmouth College  
Hanover, NH 03755

## Abstract

*Parallel computers are used to execute discrete-event simulations in contexts where a serial computer is unable to provide answers fast enough, and/or is unable to hold the simulation state in memory. Traditional research in parallel simulation has focused on the degree to which a parallel simulator provides speedup. This paper takes a different view and asks how a parallel simulator provides increased user defined utility as a result of being able to simulate larger problem sizes. We develop a model where the utility of simulating a particular simulation is an increasing function of the problem size, and ask whether overall utility accrues faster on a parallel computer if one uses it to simulate one large problem in parallel, several smaller problem instances concurrently and each in parallel, or concurrently many small problem instances on single processors. We show that under our model assumptions, utility is accrued faster either by running one large problem instance in parallel using all the available processors, or by running one small problem instance per processor, concurrently. When we consider how to optimize the utility per unit cost we find that one either runs a large problem using all available processors, multiple small problems with one per processor, or a small problem using exactly one processor. Determination of the optimal configuration depends on the user's assessment of how rapidly utility grows with the problem size. Our main contribution is to show the linkage between the effectiveness of parallel simulation and a user's perception of the value of larger problem sizes. We show that if that utility grows less than linearly in the problem size, then use of parallelism is sub-optimal. We give precise relationships between our model parameters that govern when parallelism optimizes utility, and when it optimizes price-performance. We see that when model parameters are in a "normal" range, a user's perception of utility must grow significantly—e.g. proportional to problem size raised to the 1.5<sup>th</sup> power—for parallel processing to optimize cost performance.*

## 1 Introduction

Parallel simulation has been a research topic of interest for over twenty years. Most of the work in the area is motivated by the desire to execute simulations faster, and most evaluations of parallel simulations are based on how much faster a given problem is simulated using parallelism than it is on a serial computer. There is an important alternative view in which parallel processing is a means of simulating larger problem instances than are possible on a serial computer. The speedup of the simulator is still important, but as a means of making feasible the evaluation of the larger problem, not as of an end to itself. This shift in focus is better in line with simulation user requirements, e.g. an ability to assess the end-to-end performance capacity of a nation-wide virtual private overlay network. The driving force there is the size and complexity of the network, not simply the speed at which a simulator runs. Having said that, high performance through parallelism is what *enables* analysis of the large network. Size and speed are inseparable, but the value to the user is frequently expressed as a function of the problem being simulated.

We are interested in the tension between the added benefit a user perceives of simulating increasingly larger models, and the costs of doing so. By modeling how the objective costs grow as a function of model size, we can assess how the subjective benefits must grow in order to make simulation of large models cost-effective.

When we ask how best to utilize resources such as time, and/or cost, one approach is to "discount" the benefit by the volume of resource needed to achieve that benefit. For example, we discount the utility of simulating a model of a given size by dividing that utility by the execution time required to simulate the model; we may also discount by the cost of simulating that model. Execution time increases at least in proportion to model size (or even larger), which means that for a large model to have greater discounted utility than a small one, utility as a function of model size has to grow faster than does the discounting. In economic terms this means that utility, as a function of model size, must have a larger marginal return than the marginal cost of dis-

counting. This particular observation is true in the most general of cases—for parallel simulation to make sense in the context of user utility, utility as a function of model size has to grow faster—sometimes significantly faster—than the execution time increases of simulating larger models, and/or the cost increases of simulating larger models on parallel computers.

We are able to more deeply explore this tension by using specific mathematical forms for the utility and the costs. The forms we choose are capable of representing a wide spectrum of behaviors, are simple, and are tractable from the point of view of optimization theory. Using them we come to what at first glance may seem as surprising, even counter-intuitive, results : there is no middle ground. Discounted user utility is optimized by being as aggressive as possible with parallelism, or is optimized by eschewing parallelism altogether. The value of a simple expression of model parameters determines the optimal outcome. However, as we will show, the extremity result is less surprising when one compares these two options when the determining expression is close to its decision threshold : there is actually relatively little difference between them. The value of this expression is like a single knob on the space of problem parameters. Twisting it one goes from parameter sets that strongly favor one extreme, though a region where the utility of both extremes (and possibilities in between) are much alike, to a region where the other extreme is strongly favored. The key importance of this result is in showing how the problem parameters, and the problem characteristics they quantify, play off against each other in determining whether parallelism offers the best discounted utility to the user.

Our results are not necessarily practical; decisions about utility of model size are atypical. Nor are they applicable in contexts where meeting a real-time processing deadline is the most important factor. In that case the utility of model size is subsumed by the utility of meeting real-time requirements. However our results may *explain* in part why parallel simulation is not widely adopted now, despite the availability of mature parallelizing simulators. Viewed this way, the models of user utility we present reflect unconscious attitudes towards the benefit of large models and parallelism, rather than constructively conscious ones. As is frequently the case with theory, the value is more in qualitative explanation than in quantitative prediction.

## 2 Related Work

Utility theory provides a way of describing subjective value from choices. It considers a scalar-valued “utility function” over the multi-dimensional decision space. Given decision vectors, one compares them by comparing their utility function values.

The literature on utility theory is large. Construction of utility functions is an oft-considered problem, e.g., [1, 4, 6]. Issues associated with including uncertainty in the outcome of the decision form another branch in the field. See [2] for a collection of classical papers on utility theory.

Work related to ours[3] asks whether using parallel computers to execute a simulation is cost-effective. Cost-efficiency is a specific example of utility; a validated performance model is developed to assess the cost-performance tradeoffs of increasing memory size and processing capability in a parallelized simulation. Our work is different in that we allow for user preference to define the value of running a simulation problem of a particular size. Indeed the most interesting results in our paper address how strongly a user must feel his utility grows as problem size grows, if parallelism is to be a cost-effective option.

We have used utility theory to analyze the tradeoffs between speed, memory use, and functionality among network simulators[10]. The idea in the present paper of associating utility with model size and execution time was first developed in that context. However, the earlier paper was focused on characteristics of different simulators, while the present paper is focused on the question of whether/when exploiting parallelism gives the most utility.

The nature of the results we develop are reflective of results developed in the context of load-balancing[5, 9] which showed that under “random” models of workload and communication, the optimal mapping of workload to processors was extremal—either map it all to one processor or map it evenly among all processors.

## 3 Model

Recognizing that the value of larger problems is user-dependent, our approach is based on the notion of user-defined utility. We suppose that problem size can be parameterized into “problem units” (e.g., traffic load in a network simulation, or number of queues in a queuing network simulation), and will describe problem size by a variable  $m$ ; we will use  $\mu(m)$  to denote the user’s utility of simulating an experiment with problem size  $m$ . This size is discrete by nature, but we will treat it as continuous and differentiable. This assumption does not affect the results so much as it allows concise arguments leading to those results.

In our model we wish to simply capture the notion that user utility may grow as the problem size simulated grows. A simple model that expresses a wide range of growth behaviors is  $\mu(m) = c_m m^\alpha$ , for some positive constants  $c_m$  and  $\alpha$ . Exponent  $\alpha$  expresses how rapidly utility grows as problem size grows; it turns out to be a key determinant of the optimal system configuration.

However, it is often the case that with larger problem sizes comes the need to advance the problem further into

simulation time, in order to push the system in “equilibrium” where meaningful statistics can be collected. So with increasing problem size comes the need to do more-than-proportional amount of work. This implies a trade-off problem—is the added utility of simulating large problems significant enough to offset the added computational cost of doing so, a computational cost that grows because the problem size grows *and* must be simulated further along the simulation time axis than smaller problems?

Suppose we have a parallel machine with  $N$  processors, each able to simulate up to  $m_x$  units of problem. We don’t specify whether the machine’s memory is shared or distributed. However we do assume that as the number of processors grows, the total amount of available memory grows proportionally. There are a variety of ways we might use this system to execute simulations. One extreme is to use each processor concurrently on individual problems, none greater than size  $m_x$ . Another extreme is to use the entire machine in parallel on one problem, of size no greater than  $Nm_x$ . We could also partition the machine into submachines of various sizes, and run differently sized problems in parallel on each, concurrently. We would like to be able to compare these different options, to determine which provides the greatest aggregate utility.

Each simulation experiment that is run provides a certain utility; we normalize all these options by dividing the utility a simulation run provides by the execution time required to conduct the experiment. We assume that the partitioning of the parallel system is static over a long period, and that a partition element continuously simulates a problem of the same size—as soon as one experiment of that problem size is finished, another begins. We can therefore associate with each partition element a *utility rate*, obtained by dividing the utility gained by one experiment of the chosen problem size by the execution time needed to complete the experiment. Then to get the system’s aggregate utility rate we sum together the utility rates of all its partition elements. Under these assumptions we can compare two very different configurations by comparing their respective aggregate utility rates.

We can extend this approach further and associate a *cost* with the use of a parallel machine, a cost that varies with the number of processors used. We are then interested in the system size ( $N$ ) and configuration that optimizes the *utility rate cost performance*—the aggregate utility rate per unit cost.

We approach these optimization problems by using an intuitive model of execution time which expresses a dependence on problem size and number of processors used. Under its assumptions we show that the configuration which maximizes the aggregate utility rate is always extreme: it is either best to simulate one problem using all processors, or to simulate one problem per processor, concurrently. De-

termination of which extreme is best depends on the rate of utility increase ( $\alpha$ ) in problem size, the rate ( $\epsilon$ ) at which length of the simulation must grow to reach equilibrium as the problem size grows, and the rate of performance increase ( $\beta$ ) as additional processors are used in the simulation. Of these only  $\alpha$  is subjective, and the user’s perception of how utility increases in problem size effectively determines which of the extreme configurations optimizes the aggregate utility rate.

We suppose that the cost of using a machine with  $N$  processors is proportional to the execution time multiplied by  $N^\rho$ , for some  $\rho > 0$ . This models a range of pricing policies for shared use of a large-scale shared machine. Since large-scale runs are harder to schedule, economic disincentives may be introduced for large runs, to limit them to those who need and can afford them. For this case  $\rho > 1$  is appropriate. It may alternatively be that use of large numbers of processors is actually encouraged, perhaps to better justify the original purchase of the machine. For this case a less-than-proportional cost increase is modeled using  $\rho < 1$ . For any value of  $\rho > 0$ , we show that the configuration that optimizes utility rate per unit cost is likewise extreme: either one should not use parallelism at all, or should use all of the available processors. Once again the prime determinant is the user’s subjective rate  $\alpha$  describing how utility grows with increasing problem size.

We describe native application behavior by two characteristics. The first characteristic is model size, parameterized by  $m$ , as described earlier. The second characteristic is the length of simulation time needed for an “interesting” run for a problem of size  $m$ , denoted by  $T(m)$ . In some contexts  $T(m)$  may be independent of  $m$ , in others it may increase as a function of  $m$ . For example, some classes of Markov chains are known to converge to equilibrium “exponentially fast”[11], e.g.,  $T(m) \sim \log m$ . Other Markov chains exhibit longer time requirements, even proportional to the “size” of the physical structure giving rise to the Markov chain [7].

We describe the capabilities of the simulation engine by two characteristics. The first is parameter  $\gamma$ —the average execution time needed to evaluate one unit of problem for one simulation second on one CPU of the system.  $\gamma$  may depend on  $m$ . For example, the execution cost per unit problem in a simulation where particles interact may grow in the square of the problem size, to account for analysis of all interactions. We denote the dependence of both the simulation length and the execution time per unit problem by modeling the execution time on one processor as

$$\begin{aligned} x(m, 1) &= c_t(\gamma m^{\epsilon_1}) \times m \times m^{\epsilon_2} \\ &= c_t \gamma m^{1+\epsilon} \end{aligned}$$

where we can combine the growth in  $\gamma$  and the growth in simulation run length with  $\epsilon = \epsilon_1 + \epsilon_2$ . The second char-

acteristic describes the ability of the simulation to be parallelized. We let function  $a(N)$  be the speedup of executing a problem with on a parallel system, using  $N$  processors. Our expressions for  $a(N)$  assume that the workload is evenly balanced among the  $N$  processors. In practice  $a(p)$  depends both on the application’s inherent parallelism, and on the architecture of the engine on which it runs. We use a simple descriptive model to capture the expectation that speedup is sub-linear in  $N$ . We let  $a(N) = N^\beta$  for some  $\beta \in (0, 1)$ , for all  $N \in [1, N_x]$ . We limit the range of processors to account for the expectation that *eventually* scalability issues cause performance to decrease by adding processors. The proposed model accounts for behavior over the range when adding processors improves performance. The remainder of the paper tacitly assumes that the system size is limited by  $N_x$ .

Using these concepts we express the execution time of an application with problem size  $m$  on  $N$  processors as

$$\begin{aligned} x(m, N) &= \frac{x(m, 1)}{a(N)} \\ &= \frac{c_t \gamma m^{1+\epsilon}}{N^\beta}. \end{aligned}$$

Our model supposes that a user’s *utility* of running a simulation experiment on a problem of size  $m$  is a function  $\mu(m) = c_m m^\alpha$ , for some constants  $c_m > 0$  and  $\alpha \geq 0$ . Units and interpretation of the meaning of this function are entirely the user’s. Under this model the rate (in wall-clock time) at which utility is accrued simulating a problem of size  $m$ , using  $N$  processors is given by the following definition.

**Definition 1.** *The utility rate at which utility is accrued simulating a problem of size  $m$ , using  $N$  processors, is*

$$\begin{aligned} \lambda_u(m, N) &= \frac{\mu(m)}{x(m, N)} \\ &= \frac{c_m m^\alpha}{(c_t \gamma m^{1+\epsilon})/N^\beta} \\ &= K_u m^{\alpha-(1+\epsilon)} N^\beta \end{aligned}$$

where  $K_u = c_m/(\gamma c_t)$ .

The next section establishes the main results we obtain using this model.

## 4 Partitioning

The central problem we address in this paper is a partitioning problem : how do we employ the resources of a parallel system in order to optimize a user’s aggregate utility? We first illustrate the problem by example, and then develop the formal results.

### 4.1 Example

If we have a parallel system with  $N$  processors, we can use it in different ways. We might partition it so that 1/2 of the processors are used on one large problem, 1/4 of the processors are used on a problem that is half the size, and the rest of the processors are used on individual problems. Or we could chose a different partition—there are combinatorially many different possibilities. We wish to identify the one that delivers the best overall utility. This desire is what motivated our consideration of the utility rate function. Even though different partitions may execute problems of different sizes, and have different execution times, if we consider that a partition executes one problem after another, each of the same size, then all processors in system are are busy all the time. This is illustrated in Figure 1, which depicts a four processor system. The upper partition uses all four processors on one problem, of size  $4m_x$ . The middle partition uses two processors on one problem of size  $2m_x$ , and 1 processor on each of two problems of size  $m_x$ . The lower partition uses 1 processor on each of four problems of size  $m_x$ . The execution time  $x(4m_x, 4)$  of one experiment of the first partition is larger than the execution time of one experiment using two processors ( $x(2m_x, 2)$ ), which in turn is larger than four concurrent problems of size  $m_x$ ; this is due in part to the need for larger problems to run farther in simulation time, and for the sub-linear performance gain of using parallelism. Nevertheless we can compare these partitions by computing utility rates, obtained by dividing the utility gained by one experiment on one partition element by the wall-clock time needed to complete that experiment. We can compute the aggregate rate at which utility is built up by summing the utility rates of the individual partitions.

The aggregate utility rate for the fully parallel partition is  $c_m (4m_x)^{\alpha-(1+\epsilon)} 4^\beta / (c_t \gamma)$ , for the partially parallel partition it is  $(c_m (2m_x)^{\alpha-(1+\epsilon)} 2^\beta + 2c_m m_x^{\alpha-(1+\epsilon)}) / (c_t \gamma)$ , and for the serialized partition it is  $4c_m m_x^{\alpha-(1+\epsilon)} / (c_t \gamma)$ . These expressions show the centrality of the term  $\alpha - (1 + \epsilon)$  in determining utility rate. For the purposes of illustration we choose constants  $c_m = m_x = c_t = \gamma = 1$ , and  $\beta = 0.8$  (recall that speedup is modeled as  $p^\beta$ ). Figure 2 plots the utility rate as a function of  $x = \alpha - (1 + \epsilon)$ . Interesting aspects of this graph are that either the fully parallel or the fully serial partition is always optimal, and the threshold that determines which one is optimal is  $x = 1 - \beta$ . This graph also illustrates the point we made earlier—that in the region of the crossover point, *all* of the partitions have very close to the same utility rate. It is only when the expression  $\alpha - (1 + \epsilon)$  is not close to  $1 - \beta$  that there is a pronounced difference in the utility performance of the fully parallel and fully serial partitions.

There is a strong theoretical reason for the type of behavior illustrated in this graph. We turn now to formally es-

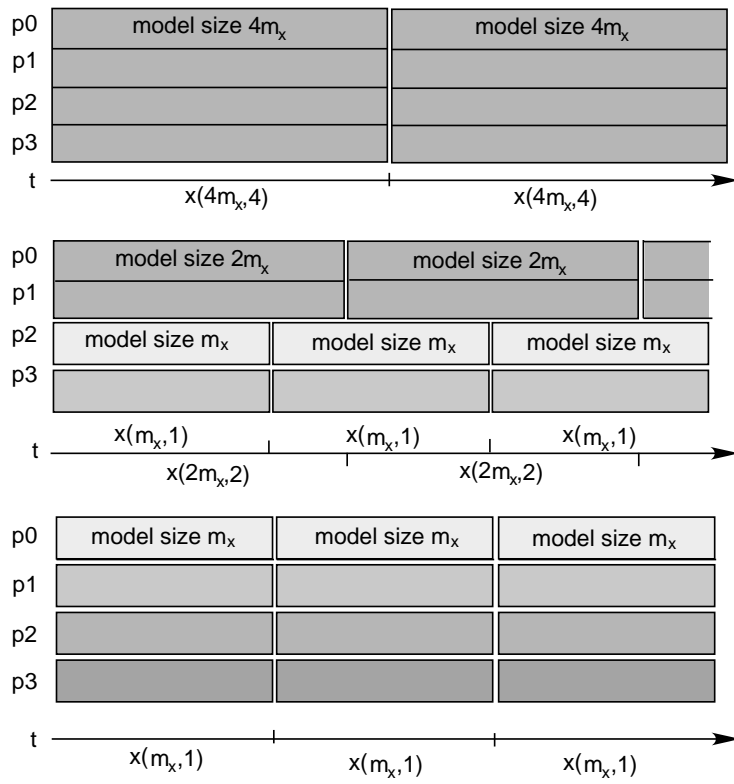


Figure 1. Measuring utility rates of two different system partitions

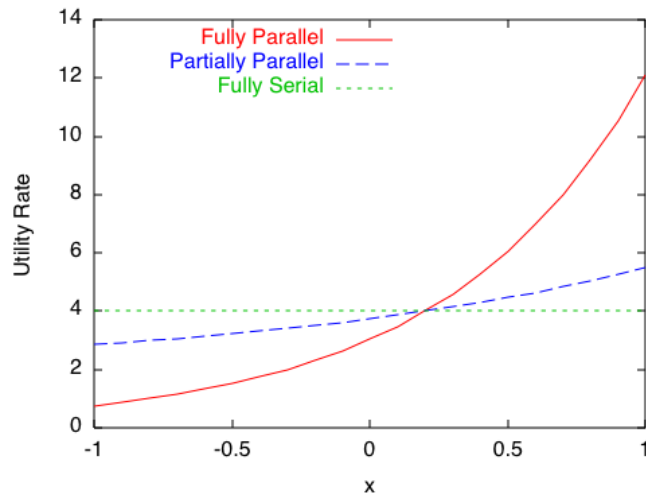


Figure 2. Illustration of utility rates for different partitions as a function of  $x = \alpha - (1 + \epsilon)$

establishing the result that the optimal partition is either fully parallel, or fully serial.

## 4.2 Analysis

Our natural notions of problem size and number of processors are discrete. However it will be convenient for us to consider them as continuous variables, so that we may take derivatives of functions like  $\lambda_u(m, N)$ . We use these derivatives legitimately to establish monotonicity and convexity properties of the discrete version of the function. Therefore it is legitimate and mechanical to derive the first and second partial derivatives of  $\lambda_u$  with respect to  $m$ . Recalling that continuous convex functions are characterized by having non-negative second derivatives, while concave functions are characterized by having non-positive second derivatives, one can establish the properties for  $\lambda_u(m, N)$  given by the following lemma.

**Lemma 1.** *Let  $\lambda_u(m, N)$  be as defined in Definition 1, and let  $N$  be fixed. Then  $\lambda_u(m, N)$  is*

- decreasing convex in  $m$  when  $\alpha - \epsilon \leq 1$ ,
- increasing concave in  $m$  when  $1 \leq \alpha - \epsilon \leq 2$ ,
- increasing convex in  $m$  when  $\alpha - \epsilon \geq 2$ .

We are also interested in the behavior of  $\lambda_u(p \cdot m, p)$  in  $p$  (whereas we considered  $N$  to be system size, we use  $p$  to denote variability), for fixed  $m$ . This describes how the aggregate utility rate increases when, as we add a processor, we likewise add  $m$  units of problem (presumably for that processor to work on). It is again mechanical to establish the properties for  $\lambda_u(p \cdot m, p)$  given by the following lemma.

**Lemma 2.** *Let  $\lambda_u(m, p)$  be as defined in Definition 1, and let  $m$  be fixed. Then  $\lambda_u(p \cdot m, p)$  is*

- decreasing convex in  $p$  when  $\alpha + \beta - \epsilon \leq 1$ ,
- increasing concave in  $p$  when  $1 \leq \alpha + \beta - \epsilon \leq 2$ ,
- increasing convex in  $p$  when  $\alpha + \beta - \epsilon \geq 2$ .

We wish also to model the financial cost of using parallel processors, in order to address configurations that optimize cost-effectiveness. Our model is based on the assumptions that cost is proportional to wallclock execution-time, and is super-linear in the number of processors used. We suppose that the cost (per unit execution time) of using  $p$  processors is  $c_c p^\rho$ , for some  $c_c > 0$  and  $\rho \geq 1$ . This models the pricing structure one expects of a shared supercomputer.

The notion of cost leads to the next definition.

**Definition 2.** *The utility rate cost performance function  $\lambda_c(m, p)$  describes the utility rate per unit cost as a function of  $p$ , and is given by*

$$\begin{aligned} \lambda_c(m, p) &= \frac{\mu(m)}{x(m, p)c_c p^\rho} \\ &= \frac{c_m m^\alpha}{c_t \gamma m^{1+\epsilon} c_c p^\rho / p^\beta} \\ &= K_c m^{\alpha-(1+\epsilon)} p^{\beta-\rho} \end{aligned}$$

where  $K_c = c_m / (\gamma c_t c_c)$ .

Since the only difference between  $\lambda_u(p \cdot m, p)$  and  $\lambda_c(p \cdot m, p)$  is the constant and the exponent of  $p$ , we can immediately characterize  $\lambda_c(p \cdot m, p)$ :

**Lemma 3.** *Let  $\lambda_c(m, p)$  be as given in Definition 2. Then  $\lambda_u(p \cdot m, p)$  is*

- decreasing convex in  $p$  when  $\alpha + \beta - \epsilon - \rho \leq 1$ ,
- increasing concave in  $p$  when  $1 \leq \alpha + \beta - \epsilon - \rho \leq 2$ ,
- increasing convex in  $p$  when  $\alpha + \beta - \epsilon - \rho \geq 2$ .

We now turn to the problem of characterizing the optimal partition. The following result is obvious, but needs to be stated.

**Lemma 4.** *Consider a problem of size  $m$  and a parallel system of  $N$  processors, and assume the constraint that every processor must work on some part of the problem. If the simulation execution time is minimized when each processor handles  $m/N$  problem units, then both the utility rate and utility rate cost performance functions are maximized when each processor handles  $m/N$  problem units.*

**Proof:** Both rate functions contain the execution time in their denominators, the rest of the functions being unaffected by workload distribution. Minimizing the execution time maximizes the rate functions.  $\square$

The next result states that when the number of processors to use is fixed, then the utility rate is maximized either for the largest problem size of interest possible on the machine, or the smallest problem of interest.

**Theorem 5.** *Let  $p$  be fixed. Suppose that every processor must simulate at least one problem unit, and that the maximum problem size that can be simulated using  $p$  processors is  $p \cdot m_x$ . Then*

- If  $\alpha + \beta - \epsilon \leq 1$ , then  $\lambda_u(m, p)$  is maximized at  $m = p$ , otherwise  $\lambda_u(m, p)$  is maximized at  $m = p \cdot m_x$ .
- If  $\alpha + \beta - \epsilon - \rho \leq 1$ , then  $\lambda_c(m, p)$  is maximized at  $m = p$ , otherwise  $\lambda_c(m, p)$  is maximized at  $m = p \cdot m_x$ .

**Proof:** By Lemma 4 we know that when the utility rate or the utility rate cost performance functions are maximized, under the constraint that each processor do some work, each processor does the same amount of work. Then the formulas of Definitions 1 and 2 apply, as do Lemmas 2 and 3, which state the thresholds for  $\lambda_u(m, p)$  and  $\lambda_c(m, p)$  to be increasing or decreasing.  $\square$

Theorem 5 has a lot to say about how model parameters interact to describe utility within the framework of our model. The beneficial effects of parallelism (reflected in  $\beta$ ) work to offset the detrimental effects of increased runtime (reflected in  $\epsilon$ ). In practice the balance will tend to weigh in favor of parallelism, as  $\beta$  will tend to be close to one, while  $\epsilon$  will tend to be much smaller (at least when the execution cost per unit problem is constant). If  $\beta - \epsilon > 0$ , then the rate of growth in utility ( $\alpha$ ) needed for large problems to have larger utility than small problems doesn't have to be even linear ( $\alpha = 1$ ). The bar is higher though when we consider the cost per unit utility. The cost exponent  $\rho$  has value 1 or larger, which means that from a cost-performance point of view, for large problems to have larger utility requires that  $\alpha + (\beta - \epsilon) \geq 1 + \rho \geq 2$ . This implies that  $\alpha$  must exceed 1, i.e. that user utility must grow super-linearly in the problem size.

We can describe the partitioning of a machine with  $N$  processors by a vector  $(p_1, p_2, \dots, p_N)$ , where  $0 \leq p_i \leq N$  for  $i = 1, 2, \dots, N$ , and  $\sum_{i=1}^N p_i = N$ . We interpret  $\lambda_u(p_i \cdot m, p_i)$  as the utility rate delivered by a submachine with  $p_i$  processors, noting that  $\lambda_u(0, 0) = 0$ . By Lemma 1 we know that when  $\alpha - \epsilon \geq 1$  then the utility rate of each individual submachine is maximized when it simulates as large a problem as it can—a submachine with  $p_i$  processors simulates a problem of size  $p_i m_x$ . In this case the aggregate utility rate delivered when the machine is so partitioned is

$$\Psi(p_1, p_2, \dots, p_N) = \sum_{i=1}^N \lambda_u(p_i \cdot m, p_i). \quad (1)$$

Still assuming that  $\alpha - \epsilon \geq 1$ , we note that since  $\beta > 0$  then  $\alpha + \beta - \epsilon > 1$ , which ensures (by Lemma 2) that  $\lambda_u(p \cdot m_x, p)$  is increasing in  $p$ . By that same lemma,  $\lambda_u(p \cdot m_x, p)$  is concave in  $p$  if  $\alpha + \beta - \epsilon \leq 2$ , and is convex in  $p$  if the inequality is reversed. In the former case  $\Psi$  is Schur-concave[8], and is maximized when every partition element has one processor; in the latter case it is Schur-convex[8] and is maximized when one partition element is given all  $N$  processors. This result is summarized below.

**Lemma 6.** *Let  $(p_1, p_2, \dots, p_N)$  describe a partition of a parallel machine, so that*

$$\Psi(p_1, p_2, \dots, p_N) = \sum_{i=1}^N \lambda_u(p_i \cdot m, p_i)$$

*gives the aggregate utility rate associated with the partition, when each processor operates on fixed  $m$  units of problem. Suppose that  $\alpha - \epsilon \geq 1$ . Then*

- *when  $\alpha + \beta - \epsilon \leq 2$  the aggregate utility rate is maximized when  $m = m_x$ , using partition  $(1, 1, \dots, 1)$ , i.e. each processor concurrently runs one problem of size  $m_x$ ;*
- *when  $\alpha + \beta - \epsilon \geq 2$  the aggregate utility rate is maximized when  $m = m_x$ , using partition  $(N, 0, 0, \dots, 0)$ , i.e. by running one problem of size  $Nm_x$  in parallel, using all  $N$  processors.*

This result says that it isn't enough just for user utility to increase in  $m$  (i.e., that  $\alpha - \epsilon \geq 1$ ). In order for parallel processing *on a single problem* to deliver higher utility than concurrent processing of parallel individual problems, we need for  $\alpha + \beta - \epsilon \geq 2$ . Since  $\beta \leq 1$  and  $0 \leq \epsilon$ , in every case we must have  $\alpha \geq 1$ . The longer that simulation runtimes grow with increasing problem size (i.e. increasing  $\epsilon$ ), and the less efficient parallel processing is (i.e. decreasing  $\beta$ ), the larger  $\alpha$  must be to satisfy this relationship.

Suppose instead that  $\alpha - \epsilon < 1$ . We know from Lemma 1 that under the constraint that each processor simulate at least one unit of problem, the utility rate is maximized when each processor simulates only one unit of problem—a partition with  $p_i$  processors simulates a problem of size  $p_i$  units. Now consider how such a partition's utility rate behaves as a function of  $p$ . That rate is

$$\lambda_u(p, p) = \frac{c_m}{\gamma c_t} p^{\alpha - (1 + \epsilon) + \beta}.$$

From this we see that  $\lambda_u(p, p)$  is monotone in  $p$ —increasing when  $p$ 's exponent is non-negative ( $\alpha - (1 + \epsilon) + \beta \geq 0$ ) and decreasing when non-positive. Under the assumption that  $\alpha - \epsilon < 1$  either case is possible, depending on  $\beta$ . However we do know that in both cases  $\alpha - (1 + \epsilon) + \beta \leq 1$ , because  $\beta < 1$ . This means that when the exponent of  $p$  is non-negative,  $\lambda_u(p, p)$  is concave in  $p$ , so that  $\sum_{i=1}^N \lambda_u(p_i)$  is Schur-concave, maximized when  $p_i = 1$  for  $i = 1, 2, \dots, N$ . When the exponent of  $p$  is negative, then  $\lambda_u(p, p)$  is decreasing in  $p$ , and is optimized at  $p = 1$ . In this case also the aggregate utility rate is achieved when each partition has exactly one processor. Thus we have proven the next lemma.

**Lemma 7.** *Let  $(p_1, p_2, \dots, p_N)$  describe a partition of a parallel machine, so that*

$$\Psi(p_1, p_2, \dots, p_N) = \sum_{i=1}^N \lambda_u(p_i \cdot m, p_i)$$

*gives the aggregate utility rate associated with the partition, where each processor operates on fixed  $m$  units of problem.*

Suppose that  $\alpha - \epsilon < 1$ . Then the aggregate utility rate is maximized when  $m = 1$ , using partition  $(1, 1, \dots, 1)$ , i.e. each processor concurrently runs one problem of size 1.

This result demonstrates that if the utility growth rate  $\alpha$  isn't at least linear in the problem size (i.e.  $\alpha = 1$ ) then parallel simulation of a large model does not make sense from the point of view of optimizing user utility.

These last two lemmas establish the basic result.

**Theorem 8.** *The partition optimizing the aggregate utility rate is extremal. When  $\alpha + \beta - \epsilon \geq 2$ , then the aggregate utility rate is maximized by simulating one problem of size  $Nm_x$  in parallel on all  $N$  processors. Otherwise the aggregate utility rate is maximized by concurrently simulating  $N$  problems, one per processor. In this case when  $1 + \beta \leq \alpha - \epsilon + \beta \leq 2$  each processor handles  $m_x$  problem units, but when  $\alpha - \epsilon < 1$  each processor handles only 1 problem unit.*

It is now not difficult to show that the configurations that optimize the utility rate cost performance are also extremal. Since the utility rate cost performance function is the aggregate utility rate divided by  $c_p N^\rho$ , when  $N$  processors are used, the configurations where the cost performance function is optimized necessarily are configurations that optimize the aggregate utility rate. Then, if the cost per processor decreases in  $N$  ( $\rho < 1$ ) then obviously the optimal solution will use as many processors as possible, If the cost per processor increases in  $N$  ( $\rho > 1$ )

**Theorem 9.** *Let*

$$\Omega(p_1, p_2, \dots, p_N) = \frac{\Psi(p_1, p_2, \dots, p_N)}{c_p N^\rho}$$

denote utility rate cost performance function, where  $c_p > 0$  and  $\rho \geq 1$ . Then the partitions maximizing  $\Omega$  are extremal :

- When  $\alpha + \beta - \epsilon - \rho \geq 1$ , then  $\Omega$  is maximized when  $m = m_x$ , using partition  $(N, 0, 0, \dots, 0)$ , and furthermore is an increasing function of  $N$ .
- When  $\alpha + \beta - \epsilon - \rho < 1$  and  $\rho \geq 1$ , then  $\Omega$  is a decreasing function of  $N$ , and so cost performance is optimized using a single processor. In this case
  - when  $\alpha - \epsilon \geq 1$  the problem simulated has  $m_x$  problem units,
  - when  $\alpha - \epsilon < 1$  the problem simulated has only 1 unit.
- When  $\alpha + \beta - \epsilon - \rho < 1$  and  $\rho < 1$ , then  $\Omega$  is an increasing function of  $N$ . Cost performance is optimized by running individual models serially. In this case

- when  $\alpha - \epsilon \geq 1$  the problem simulated on each processor has  $m_x$  problem units,
- when  $\alpha - \epsilon < 1$  the problem simulated on each processor has only 1 unit.

**Proof:** First consider the case where  $\alpha - \epsilon + \beta - \rho \geq 1$ . This implies that  $\alpha - \epsilon + \beta \geq 1 + \rho \geq 2$ , so (by Theorem 8)  $\Psi$  is optimized on a problem of size  $Nm_x$  running in parallel on  $N$  processors. We may write

$$\begin{aligned} \Omega(N, 0, 0, \dots, 0) &= \frac{c_m (Nm_x)^\alpha N^\beta}{c_t \gamma (Nm_x)^{1+\epsilon} c_c N^\rho} \\ &= K'' N^{\alpha - (1+\epsilon) + \beta - \rho} \end{aligned} \quad (2)$$

for an appropriate constant  $K''$ . Since  $\alpha + \beta - \epsilon - \rho \geq 1$ , the exponent of  $N$  is non-negative, which implies that the cost performance function is an increasing function of  $N$ .

Next consider the case where  $\alpha - \epsilon + \beta - \rho < 1$ . Depending on  $\rho$ , it is possible also for  $\alpha + \beta - \epsilon \geq 2$  (so that by Lemma 6 the utility rate is maximized on a problem of size  $Nm_x$ , using all  $N$  processors): we need for  $2 \leq \alpha + \beta - \epsilon \leq 1 + \rho$ . Equation 2 applies, but the exponent of  $N$  is negative. This means that from the point of view of utility rate cost performance, smaller is better. The best case is when  $N = 1$ —there is no parallelism at all, and the optimal model size is  $m_x$ . If instead  $\alpha + \beta - \epsilon < 2$  while  $\alpha - \epsilon \geq 1$ , then Lemma 6 says that  $\Psi$  is optimized on  $(1, 1, \dots, 1)$ , with  $m_x$  units of problem on each processor. In this case

$$\begin{aligned} \Omega(1, 1, \dots, 1) &= N \frac{c_m m_x^\alpha}{c_t \gamma c_c N^\rho} \\ &= \frac{c_m m_x^\alpha}{c_t \gamma c_c} N^{1-\rho}. \end{aligned}$$

When  $\rho \geq 1$ , this function is non-increasing in  $N$ , and so again the best case is when  $N = 1$ , and the model size is  $m_x$ . However, when  $\rho < 1$ , the function is increasing in  $N$ , but the model size simulated on each processor is  $m_x$ . The final case to consider is when  $\alpha + \beta - \epsilon < 2$  while  $\alpha - \epsilon < 1$ . Lemma 7 says that  $\Psi$  is optimized on  $(1, 1, \dots, 1)$ , with one unit of problem on each processor. The expression of  $\Omega$  for this case is identical to the one immediately above, with 1 substituted for  $m_x$ . This means that with  $\rho \geq 1$ ,  $\Omega$  is again non-increasing in  $N$ , is optimized at  $N = 1$ , where a problem of size 1 is simulated. With  $\rho < 1$ ,  $\Omega$  is increasing in  $N$ , but each processor simulates one model of size 1.  $\square$

It is interesting to note that it is possible for the utility rate to be optimized using parallelism, but have the utility rate cost performance be optimized on a serial machine. The conditions where this holds are that  $\alpha + \beta - \epsilon \geq 2$ , and  $\alpha + \beta - \epsilon - \rho \leq 1$ ; this is equivalent to

$$2 \leq \alpha + \beta - \epsilon \leq 1 + \rho.$$

Not surprisingly, the larger the growth in cost is, the easier it is to satisfy this inequality. However, when  $\rho$  is close to 1, the condition arises only when the other factors are in a delicate balance.

Theorem 9 indicates that from a utility cost performance point of view, it is necessary that  $\alpha + \beta - \epsilon - \rho \geq 1$  for parallel processing to make sense. This has potentially strong implications for what the user's sense of utility growth in problem size must be. For example, if the simulation run length grows modestly, in proportion to  $m^{0.3}$  (i.e.  $\epsilon = 0.3$ ), if parallelism delivers much—but not all—of its potential ( $\beta = 0.9$ ), and the cost of using  $N$  machines is proportional to  $N^{1.1}$ , then  $\alpha$  must be at least 1.5 for parallelism to be cost-effective. Even in the most favorable case possible ( $\beta = 1, \epsilon = 0$ ), parallel processing makes cost performance sense only when  $\alpha \geq \rho \geq 1$ , in short, utility growth must be super-linear.

## 5 Conclusion

We have examined the question of when parallel simulation is advantageous to a user, employing the notion of the user's utility. We construct an analytic model that is focused on the user's utility as a function of problem size. Our model includes factors such as super-linearly increasing workload in problem size (due to longer simulation runs, and increased work per unit problem), the inefficiencies of parallel processing, and the time-space cost of using a shared parallel system. To our knowledge our application of utility theory to this problem is unique; our results contribute both to the economic decision theory literature, and to the parallel processing literature.

We consider the problem of how to partition a parallel machine so as to maximize the aggregate rate at which user utility is accrued. Under our model assumptions we show that utility is optimized under extremal conditions. Either (i) one should simulate the largest problem possible, in parallel, using all available processors, or (ii) one should simulate the largest problem one processor can handle, concurrently on each processor in the system, or (iii) one should simulate the *smallest* problem, concurrently on each processor in the system. Furthermore, we show that if utility does not grow at least as fast as linearly in the problem size, then simulating a large model in parallel is sub-optimal.

We then consider how to optimize the utility rate cost performance function—the utility rate per unit cost. We again see an extremal solution: either use all processors in parallel on the largest problem possible, run the largest problem possible for one processor on each processor concurrently, or use only one processor (either on the largest problem that that processor can handle, or just on one unit of problem). The requirements on utility growth for the parallel approach to optimize cost performance are even higher;

it is possible to have a system where the utility rate is optimized using the parallel solution, but the cost performance is optimized using a single processor. “Typical” values for our model parameters suggest that the utility must grow in proportion to  $m^{1.5}$  or higher,  $m$  denoting the problem size.

While the modeling assumptions are simple and not especially quantitatively predictive, the *qualitative* take-home message is quite clear: when we view the usefulness of parallel simulation through the lens of problem-sized-based user utility, parallel simulation makes sense only when that utility grows *significantly* as a function of problem size.

Our work is better at explaining one reason why parallel simulation has not been widely adopted, then it is at aiding a practicing simulationist. Nevertheless, it provides a novel framework for understanding the factors that may affect user perceptions of parallel simulation. Even so, it should be noted that this conclusion rests on model assumptions which, when changed might alter the extremity conclusions, or the conditions under which parallel simulation becomes attractive. One aspect that is not captured by our model is the fact that increasing model size for a fixed number of processors tends to increase parallel efficiency—in our model efficiency is independent of model size. Our optimality results are limited to the assumption that a fixed partition set will be repeatedly used, forever. It may be that in a finite-horizon problem formulation an optimal approach would not be extremal. Further work in this area will address such issues.

## References

- [1] F. Bacchus and A. Grove. Utility independence in a qualitative decision theory. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning*, pages 542–552, Los Altos, 1996. Morgan Kaufmann.
- [2] D. Bell, H. Raiffa, and A. T. (Editors). *Decision Making: Descriptive, Normative, and Prescriptive Interactions*. Cambridge University Press, New York, New York, 1988.
- [3] B. Falsafi and D. Wood. Modeling cost/performance of a parallel computer simulator. *ACM Transactions on Modeling and Computer Simulation*, 7(1):104–130, January 1997.
- [4] W. Gorman. The structure of utility functions. *Review of Economic Studies*, 35:367–390, 1968.
- [5] B. Indurkha, H. Stone, and L. Xi-Cheng. Optimal partitioning of randomly generated distributed programs. *IEEE Transactions on Software Engineering*, SE-12:483–495, March 1986.
- [6] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, New York, 1976.
- [7] A. Manita. Convergence time to equilibrium for large finite markov chains. *Fundamental and Applied Mathematics*, 5(4):1135–1157, 1999.
- [8] A. Marshall and I. Olkin. *Theory of Majorization and Its Applications*. Academic Press, New York, New York, 1979.

- [9] D. Nicol. Optimal partitioning of random programs across two processors. *IEEE Trans. on Software Engineering*, 15(2):134–141, 1989.
- [10] D. Nicol. Utility analysis of network simulators. November 2002. Submitted for publication, [www.cs.dartmouth.edu/~nicol/papers/utility.pdf](http://www.cs.dartmouth.edu/~nicol/papers/utility.pdf).
- [11] H. Ross. *Stochastic Processes*. Wiley, New York, 1983.