

Utility Analysis of Network Simulators

David M. Nicol
Dartmouth College
Hanover, NH 03755

November 5, 2002

Abstract

A variety of network simulation tools are available today, each with its own functionality, memory demand and execution speed characteristics. Different tools are optimized for different purposes and different audiences. Nevertheless, significant differences in these characteristics creates the problem of qualifying user objectives under which a faster simulator that uses more memory is more “useful” than a slower simulation that uses less memory, and vice-versa. Our study is motivated by examination of performance characteristics of four simulators : **JavaSim**, **ns**, **SSFNet-Java**, and **SSFNet-C++**. Our approach considers “utility functions” that a user might develop to quantify his view of the speed/memory/functionality tradeoffs. We consider general properties such functions might have, and explore how these properties lead utility to favor one simulator over another. Finally, we apply utility theory to the four simulators that motivated the study, and observe that a balance of speed and model size capability tends to maximize performance.

1 Introduction

Network simulators are a critical tool for networking research and analysis. A plethora of network simulators now exist, each designed for a particular audience, each with its own functionality, memory demand, and execution speed characteristics. For example, we earlier conducted a performance study we conducted involving four simulators: **JavaSim**[10], **ns**[6], **SSFNet-Java**[12], and **SSFNet-C++**[5]. That study (described in [8]), was itself motivated by the study reported in [9] and [11] by the **JavaSim** architects. Our point in [8] was to revisit the first study, after making changes to the simulators to remove artifacts that obscured study of scalability, and to modify the original model and experiments slightly in order to increase workload as well as model size. Figure 1 shows one set of results. For our purposes the key things to note are

- that model size (the x-axis) has been parameterized by “number of connections”. The study considers performance as a function of a parametric family of models. Note that the graph has logarithmic scale on both axes.
- That different simulators reach memory-based limitations at different points in the model space.

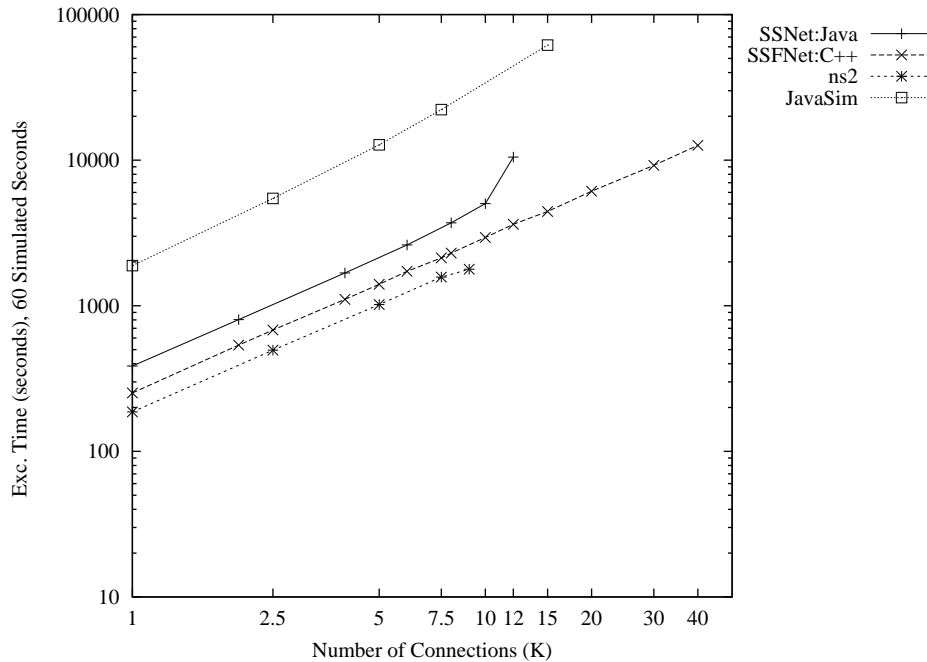


Figure 1: Execution costs of **ns**, **JavaSim**, and two **SSFNet** implementations, as a function of model size. A curve stops at the estimated maximal model size the simulator can handle.

- That execution time (the y-axis) has a different signature for different simulators.

In this graph each simulator’s curve ends when it exhausts the memory available on the simulation engine. The **JavaSim** curve is an exception. The largest model represented on the graph is a simulation of 15K connections, which required over 17 hours to complete. We ran a **JavaSim** model of size 35K connections for over two and a half weeks without it either crashing or finishing the experiment. Traces revealed that most of its time was spent in garbage collection. The same sort of behavior is seen in the curve for **SSFNet-Java**, which likewise exhibits non-linear increases in incremental execution time, which occurs as garbage collection begins to give a disproportionate contribution to overall running time.

The graph reveals tradeoffs between a simulator’s use of memory, and its execution speed. For example, **JavaSim** is ten times slower than **ns**, yet can simulate models that are twice as large. However, depending on the context, decisions about which simulator to use depend on features and functionality, not just execution speed and size of model. For example, the large memory footprint of **ns** is a result of the capability it provides to a modeler to access (through Tcl) virtually any state variable of a simulation that might be interesting. The slow execution speed of **JavaSim** is a consequence (in part) of its run-time checking of all messages across all interfaces, for coherence with a “contract” between the two components on each side of that interface. Only a user can weigh the comparative value of memory size, speed, and functionality aspects of simulators, for whatever application he has in mind.

This paper places comparative evaluation of simulators in a formal setting by introducing functions that define a strict ordering of “utility” over the multidimensional space of simulator speed/memory/functionality

characteristics. We explore how general properties of utility functions tend to favor one simulator over another. Thus, while we cannot say that one simulator is *always* better than another, we *can* say that if a user’s perception of utility behaves in certain ways, a consequence is that he will tend to favor one simulator over another. We develop a simple model that relates simulator execution time with problem size, and start with two intuitive assumptions about the user’s utility function : utility does not increase with increasing execution time, and does not decrease with increasing model size. Within this framework we compare two simulators, one is faster than the other but the slower one is able to simulate larger models. We analyze a function that gives user utility as a function of the size of the model being simulated, and the length of time used to complete a simulation experiment on that model. This base utility is scaled by a “functional utility” factor to account for user preferences unrelated to speed and memory size.

The bulk of the results show that the slower simulator must meet certain necessary (but insufficient) conditions for it to deliver the better utility. The conditions are intuitive. Being slower it will have to run longer just to get the same utility as the faster simulator; being slower, it accumulates more dis-utility of longer run times, and has to be able to simulate “large enough” models to overcome the added dis-utility. On the flip side, we show that under certain utility functions the slower simulator provably *can* overcome this dis-utility, and that under other utility functions the slower simulator *always* optimizes utility, provided that its functional utility is large enough.

2 Background

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.

Problems of classical interest include the construction of utility functions from samples of user preferences, expression of tradeoffs, and points of indifference. Others include issues associated with including uncertainty in the outcome of the decision. See [2] for a collection of classical papers on utility theory.

Some research exists on the construction of utility functions[1, 4, 7]. The main issue here is the so-called “curse of dimensionality”— if a choice involves making n decisions, then the total number of unique decision vectors over which a utility function is defined grows exponentially in n , which makes explicit construction of a utility function increasingly complex. Solutions revolve around considering utility functions that are decomposable, in the sense that preferences for some decision attributes (e.g., components of the decision vector) are invariant with respect to others. When this can be done, the utility function decomposes into a combination of simpler functions which can be developed independently of each other.

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 here is a bit different from other treaties on utility. We are not so interested in developing utility functions per se as we are in identifying general characteristics of utility functions which “tend” to give utility preference to slower simulators with larger model capacity, or faster simulators with smaller model capacity. To the extent that we do consider concrete utility functions, it is primarily for illustrative purposes.

3 Model

Our discussion centers on simulators \mathcal{S}_1 and \mathcal{S}_2 . We will consider their memory and execution requirements on a family of models whose size is abstractly parameterized by variable m . We make the simplifying assumption that overall memory requirements, and execution cost per unit simulation time, are both proportional to m . Accordingly for $i = 1, 2$ we define γ_i to be the execution time per unit simulation time per unit model, and assume that the execution time needed for s units of simulation time using simulator \mathcal{S}_i on a model of size m is

$$x_i(s, m) = \gamma_i \cdot m \cdot s.$$

This model obviously does not account for the non-linear up-turn in execution time the Java based simulators experience when garbage collection costs begin to dominate performance, but that region of performance is relatively small and might simply be considered to be outside their effective operating regime.

Increasing the model size typically increases the amount of simulation time needed to acquire statistically significant results—you can't stop the simulation before *all* the elements you wish to evaluate are ready for evaluation. However, such increases in run-length will be model dependent. For simplicity we assume that each simulation experiment must run the model for T_0 units of simulation time.

We can now express the largest model size $m_i(t)$ that can be simulated in t units of execution time, as the solution (in m) of the the equation $t = x_i(T_0, m)$:

$$m_i(t) = \left(\frac{t}{\gamma_i T_0} \right).$$

For any given set of experiments considered we assume there is a maximum amount of wallclock time t_{\max} that a user is willing to wait for a simulation experiment, and that there is a minimal model size m_0 that a user is interested in simulating. We assume a maximum available memory size \mathcal{M} , define δ_i to be the memory required per unit model by \mathcal{S}_i and denote the maximum model size that \mathcal{S}_i can simulate by $m_i^{\max} = \mathcal{M}/\delta_i$.

We suppose that a user's "base utility" of finishing an experiment of model size m in t units of wall-clock time is given by a function of two variables, $\psi(t, m)$. We assume that $\psi(t, m)$ is continuous, and piecewise differentiable. If a user's utility function is discrete (e.g. due to discrete model size), any continuous piecewise differentiable interpolation function between the discretely defined points will serve. We make the basic assumptions that for fixed model size a user's utility does not increase with increasing t , nor with fixed execution time budget does it decrease with increasing m . Formally,

$$\frac{\partial}{\partial t} \psi(t, m) \leq 0 \leq \frac{\partial}{\partial m} \psi(t, m)$$

for all points (t, m) where these derivatives exist. We call any ψ that satisfies these assumptions a *regular* utility function. Note that under any regular utility function, for a given execution time budget t , utility is always maximized by simulating the largest model possible, $m_i(t)$.

We assume that there is a functional utility of using \mathcal{S}_i that is independent of execution time and model size. Such utility might describe factors such as cost, use of ease, support, portability, programming language, transparent parallelizability, and the like. Specifically we associate a constant π_i with \mathcal{S}_i , and define \mathcal{S}_i 's *overall* utility function as

$$\psi_i(t, m) = \pi_i \cdot \psi(t, m).$$

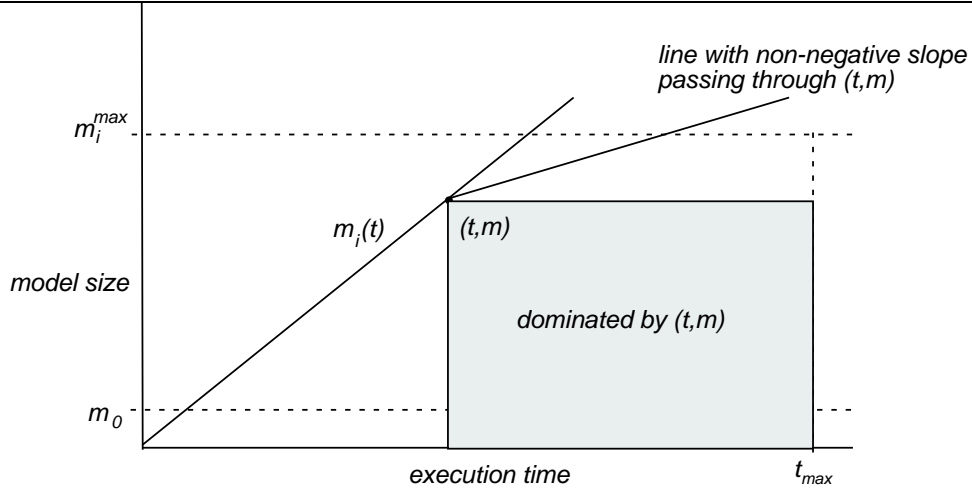


Figure 2: Simulator operational curve in utility function's domain.

Visualization of the utility function's domain is a helpful tool for understanding the intuition behind our analysis. We visualize the domain with the execution time component being the horizontal axis and the model size being the vertical axis. We envision the curve $(t, m_i(t))$ through the domain as representing the space of possible operation points for \mathcal{S}_i , each point describing one possible selection of model size and its attendant execution cost. We will call this \mathcal{S}_i 's *operational curve*, illustrated in Figure 2. Observe that \mathcal{S}_i 's operational curve is a straight line, passing through the origin, with slope $(\gamma_i T_0)^{-1}$. For any given simulator a user would seek the point on this curve where utility is maximized, and would run the experiment using the corresponding model size. Figure 2 also illustrates a point (t, m) and two geometric constructs used in our analysis. One construct is a line with non-negative slope that passes through (t, m) . We will be interested the behavior of a regular utility function constrained along such lines. The other is a shaded region which is "dominated" by (t, m) in an obvious sense. When ψ is regular, the value of ψ at any point in this region is no larger than the value of $\psi(t, m)$.

Throughout our discussion we will use the notation (t_i^{opt}, m_i^{opt}) to denote the point on \mathcal{S}_i 's operational curve where utility is maximized. Finally, without loss of generality we suppose that \mathcal{S}_1 is slower than \mathcal{S}_2 , i.e. that $\gamma_1 \geq \gamma_2$.

Table 1 summarizes our notation.

4 Utility Functions

While many of our results are general, it is helpful to consider concrete utility functions. One of these recognizes that with an expensive and shared computing system, a user's cost for running an experiment is proportional to the running time, and that for a fixed model size the utility is inversely proportional to this cost. Suppose there are constants $c_m > 0$ and $\alpha \geq 0$ such that the utility of simulating a model of size m is $c_m m^\alpha$, but that that utility is diminished in proportion to the length of time required to simulate that model.

\mathcal{S}_i	simulator $i, i = 1, 2$
$\psi(t, m)$	base utility of completing an experiment on model size m in t time
π_i	functional utility coefficient for \mathcal{S}_i
$\psi_i(t, m) = \pi_i \cdot \psi(t, m)$	\mathcal{S}_i 's overall utility function
γ_i	\mathcal{S}_i 's execution time per unit model per unit simulation time
δ_i	\mathcal{S}_i memory demand per unit model
\mathcal{M}	maximum available memory
m_0	minimum model size of interest
t_{\max}	maximum execution time of interest
m_i^{\max}	\mathcal{S}_i maximum simulatable model size
$m_i(t)$	\mathcal{S}_i maximum model size given execution time budget t
T_0	simulation time duration each experiment
$\{(t, m_i(t))\}$,	\mathcal{S}_i operational curve
$(t_i^{\text{opt}}, m_i^{\text{opt}})$	point on \mathcal{S}_i operational curve which maximizes ψ .

Table 1: Summary of notation

A utility function that recognizes the link between cost and utility is

$$\psi^{(1)}(t, m) = \frac{c_m \cdot m^\alpha}{c_t \cdot t},$$

for some scaling factor $c_t > 0$.

A related utility function uses a general utility function $u_M(m)$ for model size, and a utility function $u_E(t)$ that expresses the utility of requiring t units of execution time to complete an experiment. The overall utility is constrained to be the smaller of these :

$$\psi^{(2)}(t, m) = \min\{u_E(t), u_M(m)\}.$$

This function is regular if both u_E and u_M are appropriately monotone, continuous and piecewise differentiable.

Another model is appropriate if the user's only concern is that the simulation experiment meet a real time deadline. Once again employing the model size utility function $u_M(m)$, this formulation says that given hard deadline t_{\max} , the utility function is

$$\psi^{(3)}(m, t) = \begin{cases} u_M(m) & \text{for } t \leq t_{\max} \\ 0 & \text{for } t > t_{\max} \end{cases}.$$

This function is not rigorously regular if one considers the domain to include $t \geq t_{\max}$, but is regular if restricted to $t \leq t_{\max}$.

Still another balances model size and execution time utility linearly. It supposes constants $c_t \geq 0$ and $c_m \geq 0$ such that overall utility is a linear combination of model size utility and execution time utility :

$$\psi^{(4)}(t, m) = -c_t \cdot t + c_m \cdot m.$$

The minus sign on c_t reminds us of the negative utility of increasing execution time. The marginal utility of increasing the execution time (or for that matter, adding an additional model unit), is constant. As a result, the only parameters we will consider are those for which this marginal utility is non-negative.

These examples exhibit various general properties; some of which are enumerated below.

Definition 1. Let ψ be a regular utility function. We say that ψ is monotone when decreasing (MWD) if, for every $\mathbf{u} = (t, m)$ and $\mathbf{v} = (t', m')$ where $t' \geq t$ and $m' \geq m$, and $\psi(\mathbf{u}) \geq \psi(\mathbf{v})$, ψ is monotone non-increasing along the vector that passes from \mathbf{u} , through \mathbf{v} , and beyond.

Definition 2. Let ψ be a regular utility function. We say that ψ is monotone when increasing (MWI) if, for every $\mathbf{u} = (t, m)$ and $\mathbf{v} = (t', m')$ where $t' \geq t$ and $m' \geq m$, and $\psi(\vec{u}) \leq \psi(\vec{v})$, ψ is monotone non-decreasing along the vector that passes from \mathbf{u} through \mathbf{v} , and beyond.

Other properties of a utility function also turn out to be important.

Definition 3. Let ψ be a utility function. We say that ψ has fast growth in m if for every $\alpha > 1$, $t > 0$, and $m > 0$,

$$\psi(t, \alpha \cdot m) \geq \alpha \cdot \psi(t, m).$$

We say that ψ has slow growth in m if this inequality is reversed for all $\alpha > 1$, $t \geq 0$, and $m \geq 0$.

We make corresponding definitions to describe the utility function's behavior with respect to time.

Definition 4. Let ψ be a utility function. We say that ψ has fast decrease in t if for every $\alpha > 1$, $t > 0$, and $m > 0$,

$$\psi(\alpha \cdot t, m) \leq \frac{\psi(t, m)}{\alpha}.$$

We say that it has slow decrease in t if this inequality is reversed.

The concrete utility functions we identified have some of these properties. First consider $\psi^{(1)}$. Choose any (t, m) in its domain and consider the line $l(s) = \rho \cdot s + m - \rho \cdot t$ which passes through (t, m) with slope $\rho \geq 0$. Then applying ordinary rules of calculus

$$\begin{aligned} \frac{d}{ds} \psi^{(1)}(s, l(s)) &= \frac{d}{ds} \left(\frac{c_m \cdot l(s)^\alpha}{c_t \cdot s} \right) \\ &= \frac{c_t \cdot s \cdot \alpha \cdot c_m \cdot \rho \cdot l(s)^{\alpha-1} - c_t \cdot c_m \cdot l(s)^\alpha}{s^2}. \end{aligned}$$

Working through the algebra, a necessary and sufficient condition for this derivative to be non-negative is that

$$\rho \cdot \alpha \cdot s \geq l(s). \quad (1)$$

$\psi^{(1)}$ behaves differently when $\alpha < 1$ than when $\alpha > 1$, so consider the the former case. The left-hand-side of (1) is a linear function of s with slope $\rho \cdot \alpha$, while the right-hand-side is linear in s with slope ρ . Choose any $s = t$ such that this inequality is true, and consider what happens as we increase s beyond t . The right-hand-side of (1) increases faster in s than the left-hand-side, and the two lines eventually intersect. The derivative is non-negative until they do, and thereafter is negative. Consequently, when $\alpha < 1$, if $\psi^{(1)}$ is decreasing along a line it will always continue to decrease along that line, which is just the definition of MWD. Now consider the case where $\alpha \geq 1$. Again choosing any $s = t$ where (1) is true, we see that as s increases beyond t the left-hand-side of (1) increases faster and the derivative remains positive for increasing s , which shows $\psi^{(1)}$ in this case to be MWI. Thus we see that if $\alpha \in [0, 1]$, $\psi^{(1)}$ is MWD, and when $\alpha \geq 1$, $\psi^{(1)}$ is MWI.

It is easy to verify with ordinary algebra that $\psi^{(1)}$ has slow growth in m when $\alpha \leq 1$, and fast growth in m when $\alpha \geq 1$; likewise that $\psi^{(1)}(\alpha \cdot t, m) = \psi^{(1)}(t, m)/\alpha$. To summarize then

Observation 1. Let $\psi^{(1)}(t, m) = c_m \cdot m^\alpha / (c_t \cdot t)$. When $\alpha \in [0, 1]$, $\psi^{(1)}$ is MWD, and has slow growth in m . When $\alpha \geq 1$, $\psi^{(1)}$ is MWI, and has fast growth in m . Also, $\psi^{(1)}$ has fast and slow decrease in t .

Now consider $\psi^{(2)}$. We have

$$\frac{d}{ds}\psi^{(2)}(s, l(s)) = \begin{cases} \frac{d}{ds}u_E(s) & \text{if } u_E(s) < u_M(l(s)) \\ \frac{d}{ds}u_M(l(s)) & \text{if } u_E(s) \geq u_M(l(s)) \end{cases}.$$

Recall that $u_M(m)$ increases in m , while $u_E(t)$ decreases in t . Therefore, if $u_E(s) < u_M(l(s))$ at $s = t$, then this inequality is true for all $s \geq t$, and $\psi^{(2)}(s, l(s)) = u_E(s)$ for all $s \geq t$. Since $\frac{d}{ds}u_E(s)$ is negative, the utility function decreases all along $l(s)$ beyond t . On the other hand, if $u_E(s) \geq u_M(l(s))$ at $s = t$, then $\psi^{(2)}(s, l(s)) = u_M(l(s))$, which increases in s , until the functions $u_E(s)$ and $u_M(l(s))$ intersect. For s beyond this point $\psi^{(2)}(s, l(s)) = u_E(s)$, which is decreasing. Thus we see that $\psi^{(2)}$ is monotone when decreasing.

Observation 2. $\psi^{(2)}$ is MWD.

Turning to $\psi^{(3)}(s, l(s))$, we see that over its domain it is monotone non-decreasing. Growth properties in m depend on $u_M(m)$.

Observation 3. $\psi^{(3)}$ is MWI. Furthermore, if $u_M(\alpha m) \leq \alpha u_M(m)$ for all $\alpha \geq 1$ and $m > 0$ then $\psi^{(3)}$ has slow growth in m , while if $u_M(\alpha m) \geq \alpha u_M(m)$ for all $\alpha \geq 1$ and $m > 0$ then $\psi^{(3)}$ has fast growth in m .

Finally, consider $\psi^{(4)}$. Choose any point (t, m) and slope $\rho \geq 0$. Let $l(s) = s \cdot \rho + m - t \cdot \rho$ denote the line through (t, m) with slope ρ . The rate of change of $\psi^{(4)}$ along $l(s)$ with respect to s is

$$\begin{aligned} \frac{d}{ds}\psi^{(4)}(s, l(s)) &= \frac{\partial}{\partial s}\psi^{(4)}(s, l(s)) + \frac{d}{ds}l(s)\frac{\partial}{\partial m}\psi^{(4)}(s, l(s)) \\ &= -c_t + \rho \cdot c_m. \end{aligned}$$

Note that the derivative along $l(s)$ is therefore constant, implying that the utility function either increases or decreases monotonically along $l(s)$, depending on the constants. Also note that always $\alpha(-c_t \cdot t + c_m \cdot m) \leq -c_t \cdot t + c_m \cdot m$. In summary then

Observation 4. $\psi^{(4)}$ is both MWD and MWI. Furthermore, $\psi^{(4)}$ has fast growth in m .

Table 2 lists the identified properties, and the utility functions that enjoy them.

Next we turn to analysis of this model.

5 Analysis

A number of results can be developed using only general assumptions and properties of utility functions. Two results speak to limiting cases, one where utility is optimized on the smallest model of interest, the other where it is optimized on the largest model possible. Other results either assert that one simulator optimizes utility, or places constraints which, if not met, keep a simulator from optimizing utility. Thus we group the results in terms of whether it speaks to limiting behavior, or quality assurance of the faster or the slower simulator.

Property	$\psi^{(1)}$	$\psi^{(2)}$	$\psi^{(3)}$	$\psi^{(4)}$
MWI	$\alpha \geq 1$		✓	✓
MWD	$\alpha \in [0, 1]$	✓		✓
fast increase in m	$\alpha \geq 1$	$u_M(m) \geq \alpha u_M(m)$	$u_M(m) \geq \alpha u_M(m)$	✓
slow increase in m	$\alpha \in [0, 1]$	$u_M(m) \leq \alpha u_M(m)$		
fast decrease in t	✓			
slow decrease in t	✓			

Table 2: Utility function properties

5.1 Extremity Results

It turns out that utility functions that have both slow growth in m and fast decrease in t maximize utility on the smallest model sizes. Note that $\psi^{(1)} = c_m \cdot m^\alpha / (c_t \cdot t)$ falls into this class for $\alpha \in [0, 1]$.

Theorem 1. *Let ψ be a regular utility function that has both slow growth in m and fast decrease in t . Then for any simulator, utility is maximized on the smallest model size of interest.*

Proof: Choose any $\alpha > 1$, and any point $(t, m_i(t))$ on the operational curve of \mathcal{S}_i . Recalling that the operational curve is a straight line which passes through the origin, we have

$$\begin{aligned}
\psi(\alpha t, m_i(\alpha t)) &= \psi(\alpha t, \alpha m_i(t)) \\
&\leq \alpha \cdot \psi(\alpha t, m_i(t)) && \text{because } \psi \text{ has slow growth in } m \\
&\leq (1/\alpha)\alpha \cdot \psi(t, m_i(t)) && \text{because } \psi \text{ has inversely proportional decrease in } t \\
&= \psi(t, m_i(t)).
\end{aligned}$$

It follows that the point on the operational curve where utilization is maximized is on the smallest model size of interest. \square

A counter-part to Theorem 1 identifies conditions where a simulator's utility is optimized on the largest model size it can handle.

Theorem 2. *Let ψ be a regular utility function that has fast growth in m , and slow decrease in t . Then for either simulator \mathcal{S}_i , utility is maximized at the largest model size, m_i^{\max} .*

Proof: Choose any execution budget t and constant $\alpha > 1$ such that $\alpha t \leq x_i(T_0, m_i^{\max})$. Then

$$\begin{aligned}
\psi(\alpha t, m_i(\alpha t)) &= \psi(\alpha t, \alpha m_i(t)) \\
&\geq \alpha \cdot \psi(\alpha t, m_i(t)) \\
&\geq (1/\alpha)\alpha \cdot \psi(t, m_i(t)) \\
&= \psi(t, m_i(t)).
\end{aligned}$$

Therefore utility increases moving with increasing execution time budget along the simulator's operational curve, and is maximized at the largest model size possible for that simulator. \square

Theorems 1 and 2 show that proportional growth in utility as a function of model size, and inversely proportional decline in utility as a function of execution time are both critical thresholds. They establish qualitative properties of utility functions, just exactly the sort of results we're looking for when considering the choice of simulators. Theorem 1 in particular shows that when the cost of long executions cannot be compensated for by larger model sizes, then the faster simulator is to be preferred, albeit on the the smallest model of interest.

5.2 Optimality Results for the Faster Simulator

We next turn to results that identify conditions under which the faster simulator achieves optimal utility, and results which identify constraints the slower simulator must satisfy before it can achieve optimal utility. Study of the preconditions for these results yields insight into how user preferences bias utility towards speed rather than memory.

An obvious result is that if the faster simulator has all the advantages, it achieves the optimal utility.

Theorem 3. *If ψ is regular, $m_1^{opt} \leq m_2^{max}$ and $\pi_1 \leq \pi_2$, then $\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_2^{opt}, m_2^{opt})$.*

Proof: If $t_1^{opt} \leq t_2^{opt}$ then

$$\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_1^{opt}, m_2(t_1^{opt})) \leq \psi(t_2^{opt}, m_2^{opt}),$$

and we're done. If on the other hand $t_1^{opt} > t_2^{opt}$ then

$$\begin{aligned} \psi(t_2^{opt}, m_2^{opt}) &\geq \psi(t_1^{opt}, m_2(t_1^{opt})) \\ &\geq \psi(t_1^{opt}, m_1^{opt}). \end{aligned}$$

Since $\pi_2 \geq \pi_1$, it follows that $\pi_2 \cdot \psi(t_2^{opt}, m_2^{opt}) \geq \pi_1 \cdot \psi(t_1^{opt}, m_1^{opt})$. □

A more interesting set of conditions exist when the situation above is changed so that the utility function has fast growth in m , and the slower simulator has higher functional utility. In this case the faster simulator can tolerate a smaller functional utility and still maximize the overall utility, provided that the relative speed differential between the two be larger than the relative functional utility differential.

Theorem 4. *If ψ is regular, has fast growth in m , $m_1^{opt} \leq m_2^{max}$, and $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$, then $\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_2^{opt}, m_2^{opt})$.*

Proof:

$$\begin{aligned} \psi(t_2^{opt}, m_2^{opt}) &\geq \psi(t_1^{opt}, m_2(t_1^{opt})) \\ &= \psi(t_1^{opt}, \left(\frac{m_2(t_1^{opt})}{m_1(t_1^{opt})}\right) \cdot m_1(t_1^{opt})) \\ &\geq \left(\frac{m_2(t_1^{opt})}{m_1(t_1^{opt})}\right) \cdot \psi(t_1^{opt}, m_1(t_1^{opt})) \\ &= \left(\frac{\gamma_1}{\gamma_2}\right) \cdot \psi(t_1^{opt}, m_1^{opt}). \end{aligned} \tag{2}$$

Re-arranging, and multiplying both sides by (π_2/π_1) we get

$$\frac{\pi_2 \cdot \psi(t_2^{opt}, m_2^{opt})}{\pi_1 \cdot \psi(t_1^{opt}, m_1^{opt})} = \frac{\psi_2(t_2^{opt}, m_2^{opt})}{\psi_1(t_1^{opt}, m_1^{opt})} \geq \left(\frac{\gamma_1}{\gamma_2}\right) \left(\frac{\pi_2}{\pi_1}\right).$$

The right-hand-side of this inequality is at least as large as 1 when $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$. \square

Interestingly, the same conclusion holds true when ψ has fast decrease in t .

Theorem 5. *If ψ is regular, has fast decrease in t , $m_1^{opt} \leq m_2^{max}$, and $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$, then $\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_2^{opt}, m_2^{opt})$.*

Proof: Let $t^* = x_2(T_0, m_1^{opt})$ be the time needed by \mathcal{S}_2 to simulate the model where \mathcal{S}_1 takes its maximum utility. Observe that $t^* \leq t_1^{opt}$. Then

$$\begin{aligned} \psi(t_1^{opt}, m_1^{opt}) &= \psi((t_1^{opt}/t^*) \cdot t^*, m_1^{opt}) \\ &\leq \left(\frac{t^*}{t_1^{opt}}\right) \cdot \psi(t^*, m_1^{opt}) \\ &= \left(\frac{\gamma_2}{\gamma_1}\right) \cdot \psi(t^*, m_2(t^*)) \\ &\leq \left(\frac{\gamma_2}{\gamma_1}\right) \cdot \psi(t_2^{opt}, m_2^{opt}). \end{aligned}$$

This is equivalent to the inequality expressed in (2) above, and so the same manipulations from that point on lead to the same conclusions here. \square

Reflecting on Theorems 4 and 5 we see three conditions that jointly lead \mathcal{S}_2 to achieve optimal utility. First, a common precondition was that the slower simulator's memory use at its point of optimality was small enough for the faster simulator to handle—this removes one advantage the slower simulator might have over the faster simulator. A second important condition is that \mathcal{S}_1 is slower than \mathcal{S}_2 by a factor larger than it has more functional utility than \mathcal{S}_2 . This bounds the functional advantage \mathcal{S}_1 has over \mathcal{S}_2 . The third component is that the utility function change “rapidly”. In the first case the rapid growth is positive, in m , which ensures that the memory size difference at \mathcal{S}_1 's optimal operating point can be leveraged by \mathcal{S}_2 to overcome \mathcal{S}_1 's larger functional utility. In the second case the rapid change is negative, in t , which ensures that the execution time difference at \mathcal{S}_1 's optimal operating point can be leveraged by \mathcal{S}_2 to overcome \mathcal{S}_1 's larger functional utility.

Theorems 4 and 5 assumed that $m_1^{opt} \leq m_2^{max}$. Interestingly, the same conclusion follows if we replace this condition with $m_2^{opt} \leq m_2^{max}$, and in addition assume that ψ is MWD.

Theorem 6. *Let ψ be any regular utility function which is MWD and has fast growth in m . If $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$ and $m_2^{opt} < m_2^{max}$, then $\psi_1(t_1^{opt}, m_1^{opt}) \leq \psi_2(t_2^{opt}, m_2^{opt})$.*

Proof: If $t_1^{opt} \leq t_2^{opt}$, then

$$\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_1^{opt}, m_2(t_1^{opt})) \leq \psi(t_2^{opt}, m_2^{opt}),$$

and we're done. If on the other hand $t_1^{opt} > t_2^{opt}$ then

$$\begin{aligned}
\psi_2(t_2^{opt}, m_2^{opt}) &\geq \pi_2 \cdot \psi(t_1^{opt}, m_2(t_1^{opt})) \quad \text{because } m_2^{opt} < m_2^{\max} \text{ and } \psi \text{ is MWD} \\
&= \pi_2 \cdot \psi(t_1^{opt}, (\gamma_1/\gamma_2)m_1(t_1^{opt})) \\
&\geq \pi_2 \cdot (\gamma_1/\gamma_2) \cdot \psi(t_1^{opt}, m_1(t_1^{opt})) \quad \text{because } \psi \text{ has fast growth in } m \\
&\geq \pi_2 \cdot (\pi_1/\pi_2) \cdot \psi(t_1^{opt}, m_1(t_1^{opt})) \\
&= \psi_1(t_1^{opt}, m_1(t_1^{opt})).
\end{aligned}$$

□

The intuition behind this result is that when ψ is MWD and the optimal operating point for \mathcal{S}_2 is at a memory size less than its largest possible memory size, then even if \mathcal{S}_2 could simulate a model of size m_1^{opt} , its utility in doing so is less than its utility of simulating a model of size m_2^{opt} , and the difference in base utility between the two simulators at model size m_1^{opt} is big enough to overcome any advantage \mathcal{S}_1 might have in functional utility over \mathcal{S}_2 .

Intuition tells us that the slower simulator will have to run longer than the faster simulator to achieve at least as much utility as does the faster simulator (depending on the functional utility constants). Our next result confirms that intuition by demonstrating a necessary (but insufficient) condition for utility to be optimized using the slower simulator. An implication is that if t_{\max} is less than the time value identified by the theorem, then utility is maximized using the faster simulator.

Theorem 7. *Let ψ be any regular utility function which has fast growth in m . If $\pi_2 \leq \pi_1$ and $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$ then*

$$\psi_1(s, m_1(s)) \leq \psi_2(t_2^{opt}, m_2^{opt})$$

for all $s \in [0, (\pi_2/\pi_1)(\gamma_1/\gamma_2)t_2^{opt}]$. Consequently, if $t_{\max} \leq (\pi_2/\pi_1)(\gamma_1/\gamma_2)t_2^{opt}$, then $\psi_1(t_1^{opt}, m_1^{opt}) \leq \psi_2(t_2^{opt}, m_2^{opt})$.

Proof: First we show that the claimed relationship holds for all $s \in [0, t_2^{opt}]$, for

$$\begin{aligned}
\psi_2(t_2^{opt}, m_2^{opt}) &= \pi_2 \cdot \psi(t_2^{opt}, m_2^{opt}) \\
&\geq \pi_2 \cdot \psi(s, m_2(s)) \quad \text{for all } s \in [0, t_2^{opt}] \\
&= \pi_2 \cdot \psi(s, (\gamma_1/\gamma_2)m_1(s)) \\
&\geq \left(\frac{\gamma_1}{\gamma_2}\right) \cdot \pi_2 \cdot \psi(s, m_1(s)) \quad \text{because } \psi \text{ has fast growth in } m \\
&= \left(\frac{\pi_2}{\pi_1}\right) \left(\frac{\gamma_1}{\gamma_2}\right) \cdot \psi_1(s, m_1(s)) \\
&\geq \psi_1(s, m_1(s)) \quad \text{because } (\pi_2/\pi_1)(\gamma_1/\gamma_2) \geq 1.
\end{aligned}$$

Now define t^* to be the execution time budget under which \mathcal{S}_1 simulates a model which is a factor of (π_1/π_2) smaller than m_2^{opt} , e.g., that $(\pi_1/\pi_2)m_1(t^*) = m_2^{opt}$. It is straightforward to calculate that $t^* = (\pi_2/\pi_1)(\gamma_1/\gamma_2)t_2^{opt}$. Then

$$\psi_2(t_2^{opt}, m_2^{opt}) = \pi_2 \cdot \psi(t_2^{opt}, m_2^{opt}) = \pi_2 \cdot \psi(t_2^{opt}, (\pi_1/\pi_2)m_1(t^*))$$

$$\begin{aligned}
&\geq \pi_2 \cdot (\pi_1/\pi_2) \cdot \psi(t_2^{opt}, m_1(t^*)) \\
&= \psi_1(t_2^{opt}, m_1(t^*)) \\
&\geq \psi_1(s, m_1(s)) \quad \text{for any } s \in [t_2^{opt}, (\pi_2/\pi_1)(\gamma_1/\gamma_2)t_2^{opt}].
\end{aligned}$$

The last step follows from the regularity of ψ , as $t_2^{opt} \leq s$ and $m_1(t^*) \geq m_1(s)$. \square

All the other results in this section are conditioned on one or the other of the simulators achieving its optimal utility on a model size that is smaller than the largest possible. Our last result applies when both simulators optimize utility at their largest model sizes. It directly addresses the tradeoff between the decrease in utility the slower simulator may suffer to run long enough to catch the faster simulator's optimized utility, and the increase in utility that it may achieve by doing so.

For notational simplicity we define

$$r(t, m) = \frac{-\frac{\partial}{\partial t}\psi(t, m)}{\frac{\partial}{\partial m}\psi(t, m)} \quad (3)$$

The value of $r(t, m)$ is a statement of how rapidly the utility is decreasing at (t, m) due to increasing execution time, relative to how rapidly it is increasing due to increasing model size. Our interest in this ratio comes from the derivative of ψ with respect to s along a line through (t, m) with slope ρ — that derivative is negative if $\rho < r(t, m)$, and is positive otherwise, for

$$\frac{d}{dt}\psi(t, l(t)) = \frac{\partial}{\partial t}\psi(t, l(t)) + \rho \frac{\partial}{\partial m}\psi(t, l(t)).$$

Using $r(t, m)$, we can identify situations where the added utility of simulating a larger model does not offset the added execution cost of doing so.

Lemma 8. *Let ψ be a utility function that is MWD. Suppose that $m_1^{\max} \geq m_2^{\max}$. If*

$$\frac{m_1^{\max} - m_2^{opt}}{x_1(T_0, m_1^{\max}) - x_2(T_0, m_2^{opt})} \leq r(x_2(T_0, m_2^{opt}), m_2^{opt}),$$

then $\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_2^{opt}, m_2^{opt})$. Consequently, if $\pi_2 \geq \pi_1$, then $\psi_1(t_1^{opt}, m_1^{opt}) \leq \psi_2(t_2^{opt}, m_2^{opt})$.

Proof: $\rho = (m_1^{\max} - m_2^{\max}) / (x_1(T_0, m_1^{\max}) - x_2(T_0, m_2^{\max}))$ is the slope of the straight line which passes through $(x_2(T_0, m_2^{\max}), m_2^{\max}) = (t_2^{opt}, m_2^{opt})$ and the operational point of \mathcal{S}_1 at the largest model size it can handle. Beyond (t_2^{opt}, m_2^{opt}) , this line lies on or above the line between (t_2^{opt}, m_2^{opt}) and (t_1^{opt}, m_1^{opt}) . Since the two lines have a common endpoint, the second line has smaller slope. See Figure 3. Since the slope of the second line is less than ρ , the rate of change of ψ in the direction of the second line is negative. Since ψ is MWD it follows that $\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_2^{opt}, m_2^{opt})$. \square

This somewhat technical result is the foundation to a more compelling one.

Theorem 9. *Let R be such that $R \leq r(t, m)$ for $m_0 \leq m \leq m_1^{\max}$ and $x_1(T_0, m_0) \leq t \leq x_1(T_0, m_1^{\max})$ on points where $r(t, m)$ is defined. If $T_0 \geq 1/(R\gamma_1)$ and ψ is MWD, then $\psi(t_1^{opt}, m_1^{opt}) \leq \psi(t_2^{opt}, m_2^{opt})$.*

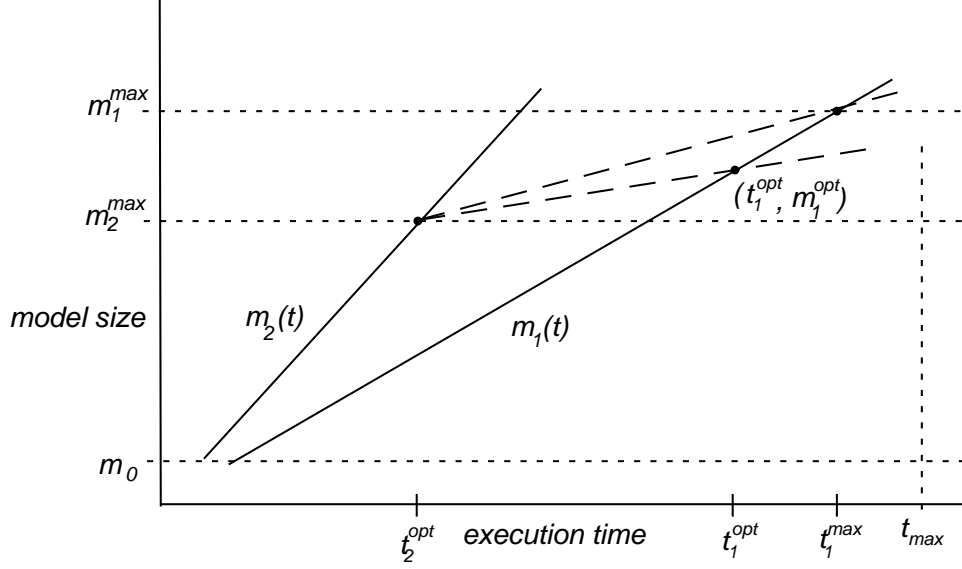


Figure 3: Geometric argument.

Proof: Since $T_0 \geq 1/(R\gamma_1)$, it can be verified that

$$\begin{aligned}
 \frac{R\gamma_2 T_0 - 1}{R\gamma_1 T_0 - 1} &\leq \frac{R\gamma_2 T_0}{R\gamma_1 T_0} \\
 &= \frac{\gamma_2}{\gamma_1} \\
 &\leq 1 \\
 &\leq \frac{m_1^{\max}}{m_2^{\text{opt}}}.
 \end{aligned}$$

One can re-arrange this inequality to find

$$\frac{m_1^{\max} - m_2^{\text{opt}}}{\gamma_1 m_1^{\max} T_0 - \gamma_2 m_2^{\text{opt}} T_0} \leq R.$$

The result follows then from direct application of Lemma 8. \square .

In the case of $\psi^{(4)}$, $r(t, m) = c_t/c_m$, a constant. Theorem 9 then implies that simply increasing the length of the simulation run eventually ensures that optimal utility is obtained using the faster simulator. A fuller explanation for this is developed in §5.4.2.

This subsection has primarily looked at conditions under which the faster simulator achieves the better utility. Conditions which crop up repeatedly are ones that signal tendencies towards assured optimality of the faster simulator. One of the signals is when one or the other of the simulators achieves its optimal utility on a model size smaller than the maximum one possible. Intuitively this suggests that the utility function structure does not emphasize model size to the degree it needs to in order for the slower simulator to overcome its speed liability. Another condition is that $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$, which is a formal way of

saying that the slower simulator may have more functional utility, but not enough to overcome its speed liability. Theorem 9 is not so much a statement of conditions favoring the faster simulator, as it is a lower bound on how much longer the slower simulator must execute than the faster simulator to have any chance of achieving better utility. Theorem 9 gives the surprising result that for a broad class of utility functions, the length of the simulation run (in simulation time) can affect the determination of which simulator delivers the highest utility.

5.3 Optimality Results for the Slower Simulator

Another set of results identify conditions under which utility is optimized by the slower simulator. For instance, intuition suggests that if the slower simulator's functional utility is large enough and/or the largest model it can simulate is large enough, then its comparative disadvantage due to speed can be overcome. We can demonstrate this in two particular cases. In the first case we assume that utility growth is slow in m . The proof exploits the existence of a natural bound on the relative difference between the slower simulator's utility and the faster simulator's utility, given the same execution time budget.

Theorem 10. *Let ψ be regular, have slow growth in m , $(\pi_1/\pi_2) \geq (\gamma_1/\gamma_2)$, and $t_1^{opt} \leq t_{max}$. Then $\psi_1(t_1^{opt}, m_1^{opt}) \geq \psi_2(t_2^{opt}, m_2^{opt})$.*

Proof: Choose any $t > 0$. As ψ is slow in m , we have

$$(\gamma_1/\gamma_2)\psi(t, m_1(t)) \geq \psi(t, (\gamma_1/\gamma_2)m_1(t)) = \psi(t, m_2(t)).$$

Consequently

$$\frac{\pi_1}{\pi_2} \geq \frac{\gamma_1}{\gamma_2} \geq \frac{\psi(t, m_2(t))}{\psi(t, m_1(t))},$$

from which it follows that

$$\psi_1(t, m_1(t)) = \pi_1 \cdot \psi(t, m_1(t)) \geq \pi_2 \cdot \psi(t, m_2(t)) = \psi_2(t, m_2(t)).$$

If, in particular we choose $t = t_2^{opt}$, we obtain

$$\psi_1(t_2^{opt}, m_1(t_2^{opt})) \geq \psi_2(t_2^{opt}, m_2(t_2^{opt})).$$

The result follows from the observation that the left-hand-side of this inequality is bounded from above by $\psi_1(t_1^{opt}, m_1^{opt})$. \square

Interestingly, the second case we find where the functional utility of the slower simulator overcomes its speed disadvantage is when ψ has fast growth in m , and the slower simulator's utility is protected by slow decrease in t . The key idea here is that if the slower simulator's operational curve can just get into the region where the slower simulator can handle models larger than the faster simulator, the combination of limited degradation due to execution delay and accelerated increase due to model size will allow a large enough functional utility to dominate.

Theorem 11. *Let ψ be a regular utility function that has fast growth in m , and slow decrease in t . If $(\pi_1/\pi_2) \geq (\gamma_1/\gamma_2)$ and $t_1^{opt} \leq t_{max}$, then for any $m^* \geq m_2^{opt}$, $\psi_1(x_1(T_0, m^*), m^*) \geq \psi_2(t_2^{opt}, m_2^{opt})$. In particular, $\psi_1(t_1^{opt}, m_1^{opt}) \geq \psi_2(t_2^{opt}, m_2^{opt})$.*

Proof: By Theorem 2 the utility of \mathcal{S}_1 is maximized on a model of size $m_1^{\max} \geq m_2^{\max}$, implying that $t_1^{\text{opt}} \geq t_2^{\text{opt}}$. Then for any $m^* \geq m_2^{\text{opt}}$,

$$\begin{aligned}
\psi(x_1(T_0, m^*)) &= \psi\left(\left(\frac{x_1(T_0, m^*)}{t_2^{\text{opt}}}\right) \cdot t_2^{\text{opt}}, \left(\frac{m^*}{m_2^{\text{opt}}}\right) \cdot m_2^{\text{opt}}\right) \\
&\geq \left(\frac{m^*}{m_2^{\text{opt}}}\right) \cdot \psi\left(\left(\frac{x_1(T_0, m^*)}{t_2^{\text{opt}}}\right) \cdot t_2^{\text{opt}}, m_2^{\text{opt}}\right) \quad \text{because } \psi \text{ has fast growth in } m \\
&\geq \left(\frac{m^*}{m_2^{\text{opt}}}\right) \cdot \left(\frac{t_2^{\text{opt}}}{x_1(T_0, m^*)}\right) \cdot \psi(t_2^{\text{opt}}, m_2^{\text{opt}}) \quad \text{because } \psi \text{ has slow decrease in } t \\
&= (\gamma_2/\gamma_1)\psi(t_2^{\text{opt}}, m_2^{\text{opt}}) \quad \text{working through the algebra.}
\end{aligned}$$

Rearranging and multiplying each side of the inequality by (π_1/π_2) yields

$$\begin{aligned}
\frac{\psi_1(x_1(T_0, m^*), m^*)}{\psi_2(t_2^{\text{opt}}, m_2^{\text{opt}})} &\geq \left(\frac{\pi_1}{\pi_2}\right) \cdot \left(\frac{\gamma_2}{\gamma_1}\right) \\
&\geq 1 \quad \text{because } (\pi_1/\pi_2) \geq (\gamma_1/\gamma_2).
\end{aligned}$$

□

The outstanding common characteristic of these two results are that $(\pi_1/\pi_2) \geq (\gamma_1/\gamma_2)$ —the comparative functional advantage of the slower simulator is larger than the comparative disadvantage of its slower speed. The other conditions in the statement of the theorem may be seen as technical, identifying conditions when functionality can dominate speed.

5.4 Specific Results

Finally we turn to results that are specific to two of the four utility functions discussed earlier. These show with more specificity some of the broad characteristics of utility functions that favor one simulator over another.

5.4.1 $\psi^{(1)}$

When $\alpha \in [0, 1]$, $\psi^{(1)}(t, m) = c_m \cdot m^\alpha / (c_t \cdot t)$ has both slow increase in m , and fast decrease in t . Theorem 1 then states that utility is optimized on the smallest problem size of interest. This suggests that the only form of $\psi^{(1)}$ of real interest is when $\alpha \geq 1$. Now if $\alpha = 1$ the value of the utility function along \mathcal{S}_i 's operational line is

$$\begin{aligned}
\psi_i^{(1)}(t, m_i(t)) &= \pi_i \frac{c_m \cdot \left(\frac{t}{\gamma_i T_0}\right)}{c_t \cdot t} \\
&= \pi_i \frac{c_m}{c_t \gamma_i T_0}.
\end{aligned}$$

This expression shows that a given simulator's utility is constant at all its operational points, and furthermore that the faster simulator's utility is higher when $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$. This proves the next result.

Theorem 12. Suppose there exist $c_m \geq 0$ and $c_t > 0$ such that $\psi_i^{(1)}(t, m) = \pi_i \cdot c_m m / (c_t t)$. If $(\gamma_1/\gamma_2) \geq (\pi_1/\pi_2)$, then $\psi_1^{(1)}(t_1^{opt}, m_1^{opt}) \leq \psi_2^{(1)}(t_2^{opt}, m_2^{opt})$.

The specific form of the utility function allows us to quantify the memory/speed/functionality tradeoff in a way that is not possible in the general case. When $\alpha > 1$ and $\pi_1 \geq \pi_2$ we know that if the slower simulator can handle a model size that is sufficiently larger than the slower simulator, that it will optimize utility. We can precisely identify that model size, as follows. Rewrite the utility function along \mathcal{S}_i 's operational curve as $\psi_i^{(1)}(x_i(T_0, m), m) = c_m \cdot m^\alpha / (c_t \cdot \gamma_i \cdot T_0 \cdot m)$. Observe that this utility is monotone increasing in m . For any given m we can ask for what multiple βm , $\beta > 1$, we have $\psi_1^{(1)}(x_1(T_0, \beta m), \beta m) = \psi_2^{(1)}(x_2(T_0, m), m)$. Memory βm is a cross-over point, on memory sizes larger than βm , the slower simulator's utility is greater than the faster simulator's utility on a model of size m . The solution of β in the equation

$$\pi_1 \frac{c_m (\beta m)^\alpha}{c_t \gamma_1 T_0 \beta m} = \pi_2 \frac{c_m \cdot m^\alpha}{c_t \gamma_2 T_0 m}$$

is $\beta = \left(\frac{\gamma_1 \pi_2}{\gamma_2 \pi_1} \right)^{1/(\alpha-1)}$. Note that β is independent of m . If $\beta < 1$, the slower simulator achieves parity with the faster simulator on a smaller model size than m_2^{\max} ; if $\beta \geq 1$, it describes precisely what the ratio of m_1^{\max}/m_2^{\max} must be for the slower simulator to achieve better utilization. These observations are encapsulated in the theorem below.

Theorem 13. Consider $\psi^{(1)}$, suppose that $\alpha > 1$, and let $\beta = \left(\frac{\gamma_1 \pi_2}{\gamma_2 \pi_1} \right)^{1/(\alpha-1)}$.

- If $\beta < 1$, then $\psi_1^{(1)}(t_1^{opt}, m_1^{opt}) \geq \psi_2^{(1)}(t_2^{opt}, m_2^{opt})$.
- If $\beta \geq 1$ and $m_1^{\max}/m_2^{\max} \leq \beta$, then $\psi_1^{(1)}(t_1^{opt}, m_1^{opt}) \leq \psi_2^{(1)}(t_2^{opt}, m_2^{opt})$.
- If $\beta \geq 1$ and $m_1^{\max}/m_2^{\max} \geq \beta$, then $\psi_1^{(1)}(t_1^{opt}, m_1^{opt}) \geq \psi_2^{(1)}(t_2^{opt}, m_2^{opt})$.

The specificity of the objective function lets us fully characterize when one simulator dominates another. From the point of view of broader-scale characteristics, we see again the importance of the relationship between (γ_1/γ_2) and (π_1/π_2) —it is possible for the slower simulator to achieve the larger utility only if its functional utility advantage is large enough to offset its speed disadvantage.

5.4.2 $\psi^{(4)}$

$\psi^{(4)}(t, m) = -c_t \cdot t + c_m \cdot m$ is a linear function. Given c_t and c_m , the derivative of $\psi^{(4)}$ along a simulator's operational curve is constant :

$$\frac{d}{dt} \psi^{(4)}(t, m_i(t)) = -c_t + \frac{c_m}{\gamma_i T_0}.$$

This expression is negative if and only if $1/(\gamma_i T_0) < c_t/c_m$. If negative it is better not to run \mathcal{S}_i at all, than to run it on the smallest model size of interest. Note the connection with Theorem 9, which asserts that optimal utility is obtained using the faster simulator, when $T_0 > 1/(R\gamma_1)$. Using $R = c_t/c_m$ this condition is rewritten as $1/(\gamma_i T_0) < c_t/c_m$.

$\psi^{(4)}$ is tractable enough for us to be able to say definitively when one simulator or the the other optimizes utility, depending on model size and relative values of the functional utility. In order to discover what a “break-even” model size must be, we ask when the utility of \mathcal{S}_1 running a model of size βm has the same utility as \mathcal{S}_2 running a model of size m . That is, we solve for β in the equation

$$\pi_1(-c_t\gamma_1T_0\beta m + c_m\beta m) = \pi_2(-c_t\gamma_2T_0m + c_m m).$$

Simple algebra establishes that

$$\beta = \left(\frac{\pi_2}{\pi_1}\right) \left(\frac{-c_t\gamma_2T_0 + c_m}{-c_t\gamma_1T_0 + c_m}\right).$$

Observe that β does not depend on m . We recognize the term $-c_t\gamma_iT_0 + c_m$ as \mathcal{S}_i ’s marginal utility of increasing the model size by one unit. If both such marginal utilities are negative, we shouldn’t be running the experiments using either simulator. If \mathcal{S}_1 ’s marginal utility is negative while \mathcal{S}_2 ’s is non-negative, then we should run the experiment using \mathcal{S}_2 . The more interesting situation is when the marginal utilities of both simulators are positive. In this case both simulators maximize their utility at the largest model size they can handle. We then have

$$\frac{\psi_2^{(4)}(t_2^{opt}, m_2^{opt})}{\psi_1^{(4)}(t_1^{opt}, m_1^{opt})} = \frac{\psi_2^{(4)}(x_2(T_0, m_2^{max}), m_2^{max})}{\psi_1^{(4)}(x_1(T_0, m_1^{max}), m_1^{max})} = \left(\frac{m_2^{max}}{m_1^{max}}\right) \beta.$$

This discussion is summarized by the following theorem.

Theorem 14. Consider $\psi_i^{(4)}(t, m(t)) = -c_t t + c_m \left(\frac{t}{\gamma_i T_0}\right)$. If $1/(\gamma_i T_0) < c_t/c_m$, then $\psi_i^{(4)}(x_i(T_0, m_0), m_0) < 0$, and it is better not to run the experiment with \mathcal{S}_i than it is to run it on the smallest model size of interest. If $1/(\gamma_i T_0) > c_t/c_m$ for both $i = 1, 2$, then $\psi_1^{(4)}(t_1^{opt}, m_1^{opt}) \leq \psi_2^{(4)}(t_2^{opt}, m_2^{opt})$ if and only if

$$\left(\frac{m_2^{max}}{m_1^{max}}\right) \left(\frac{\pi_2}{\pi_1}\right) \left(\frac{-c_t\gamma_2T_0 + c_m}{-c_t\gamma_1T_0 + c_m}\right) > 1.$$

We therefore see in this theorem a complete characterization of which simulator maximizes utility, as a function of m_i^{max} , γ_i , and π_i .

6 Application

We now analyze the empirical results we reported in [8] in light of this utility theory.

The table below reports for each of the four simulators the maximum model size (in “connections”), execution cost (in seconds per connection per unit sim-second), and inverse of the total execution time per unit model.

simulator	m_i^{max}	γ_i	$1.0/(\gamma_i * 60sec)$
SSFNet-C++	35,000	4.1 ms	4.06
ns	9,000	3.1 ms	5.37
SSFNet-Java	10,000	6.1 ms	2.73
JavaSim	15,000	41.0 ms	0.40

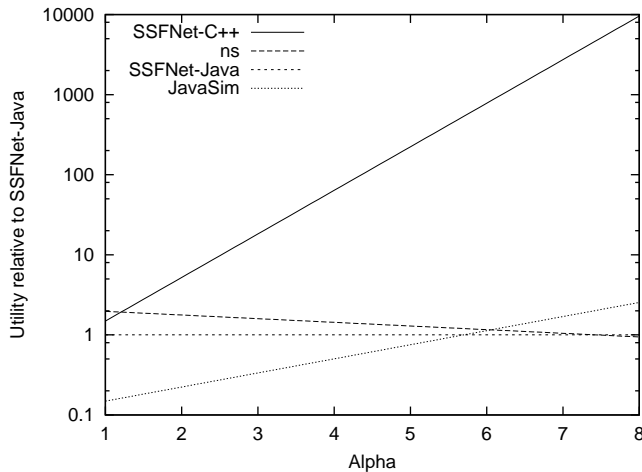


Figure 4: Utility (relative to **SSFNet-Java** using $\psi(t, m) = c_m(m^{opt})^\alpha / c_t t^{opt}$, as a function of α .

This table shows that **SSFNet-C++** simulates more model, faster, than both **SSFNet-Java** and **JavaSim**. Ignoring functional utility, under any regular utility function its utility will be largest among those three simulators. This implies that in the group of four, assuming the same functionality utility for each simulator, either **SSFNet-C++** or **ns** has the largest utility, depending on the utility function. These relations change of course, arbitrarily, with arbitrary assignment of functional utility constants to each simulator.

First consider the function $\psi^{(1)}(t, m) = c_m m^\alpha / (c_t t)$. The function’s emphasis on model size is clearly an increasing function of α . For the purposes of comparing two simulators, the values of c_t and c_m don’t matter, they cancel out when one takes the ratio of two simulators’ utility functions. Theorem 2 assures us that $m_i^{opt} = m_i^{max}$, so we compute optimal utilities based on the measured values of m_i^{max} , and $t_i^{opt} = \gamma_i \cdot T_0 \cdot m_i^{max}$. To get a sense of how the different simulators’ utilities relate to each other, we plot a graph where the **SSFNet-Java** utility is taken as a baseline, by computing the ratio of a simulator’s utility to that of **SSFNet-Java**, as a function of α . This is illustrated in Figure 4. Note that the y-axis is logarithmic. We see that the execution speed of **ns** gives it the advantage for smaller values of α , but the model size capacity of **SSFNet-C++** begins gives it dominance for larger α . For large enough values of α , the utility of **JavaSim** dominates simulators with smaller values of m_i^{max} , but at those cross-over points the utility of **SSFNet-C++** is two orders of magnitude larger.

Now suppose we evaluate utility using $\psi^{(4)}(t, m) = -c_t \cdot t + c_m \cdot m$. If we chose the units of utility so such that $c_m = 1$, any equation of the form $\psi(s, m) = \psi(s', m')$ can be solved for c_t (with the constraint that $s \leq s'$ and $m \leq m'$). One way to motivate different constants is to use the table above to assert different execution time-memory balances. We can’t expect a simulator to take less than twice as long on a model twice the size. If we say that $\psi(t, m) = \psi(2t, 2m)$ we implicitly say that utility for \mathcal{S}_i is dependent on its execution speed—we cannot accept anything less than a linear increase in model capability given a linear

increase in execution time budget.

Using this approach we create four sets of constants. The set derived from simulator \mathcal{S}_i are from the solution ($c_t = m_i^{\max}/t$) of

$$-c_t \cdot t + m_i^{\max} = -2c_t \cdot t + 2m_i^{\max}.$$

The table below gives the utility for each simulator, for each of the four sets of constants so created. Each column corresponds to a given assignment of constants; a given simulator’s utilities for different constants are read off across a row, the simulators are enumerated for reference. Entries ‘-’ denote situations where utility is optimized by not running that simulator.

Simulator/constants	from SSFNet-C++	from ns	from SSFNet-Java	from JavaSim
SSFNet-C++	0	-	11,475	31,500
ns	2195	0	4,426	8,319
SSFNet-Java	-	-	0	8,512
JavaSim	-	-	-	0

Note that the fastest simulator is not always the one that optimizes utility—note the last two columns of the table.

An alternative means of constructing constants is to assert, for each simulator, that simulating more model is important enough so that we’re willing to accept a factor of four increase in execution time to achieve a factor of two increase in model size. These constants come from solving equations of the form $\psi(t, m) = \psi(4t, 2m)$. According to this methodology, we get the following utility values:

Simulator/constants	from SSFNet-C++	from ns	from SSFNet-Java	from JavaSim
SSFNet-C++	23,333	19,569	27,158	33,833
ns	6,731	6,000	7,475	8,773
SSFNet-Java	5,040	3,441	6,666	9,504
JavaSim	-	-	-	10,000

Examination of these two tables shows how comparison of simulators strongly depends on relative weight placed on execution speed, and memory size. Notable differences include the fact that in the second set of constants, more of the simulators are eligible to run. The comparative utility of **SSFNet-C++** and **ns** are reversed on the constants based on their respective operating points. There is an intuitive correspondence here with the first set—which favors execution speed—leaning towards **ns**, while the second set—which favors model size—leans towards **SSFNet-C++**.

The utility of **Java-SSFNet** does not dominate the utility of **ns** in any of the settings displayed. Presumably one could fine-tune the constants to find a set where that occurs, to take advantage of the small difference in maximal problem size between the two.

In the first table **JavaSim** does not dominate any simulator, ever. In the second table, there is one case where its utility exceeds that of two simulators, primarily because it is able to simulate larger models than either of

those. There remains a factor of 3.38 difference though between **JavaSim** and **SSFNet-C++**, whose utility substantially dominates the others in all cases considered here.

While **SSFNet-C++** is somewhat slower than **ns**, it uses significantly less memory than **ns**. $\psi^{(4)}$ balances the loss due to slowness against the gain due to model capacity; in 4 of the 6 constants considered **SSFNet-C++** achieves the better utility.

Functional utility is subjective. However it is easy to now consider the effects of it on comparative orderings. In our model all that a functional utility constant does is scale the functional surface of ψ ; in the utility tables above one just multiplies the given utility by its simulator's functional utility constant. Therefore, when the analysis of ψ alone indicates that a simulator's utility is optimized by not running it at all, that conclusion remains the same. Likewise, in the tables above, simulators with zero utility have constant utility regardless of model size and execution budget, their scaled utilities are constant, and should be zero because no other set of constants will indicate that a simulator has non-negative utility that is less. But, given two simulators with base utilities $u_i > 0$ and $u_j > 0$ with $u_i < u_j$, we see that a necessary and sufficient condition for \mathcal{S}_i to have greater utility than \mathcal{S}_j is that $\pi_i/\pi_j > u_j/u_i$. Thus, if a user weights the utility that **SSFNet-Java** has a much larger and mature library of components than does **SSFNet-C++** by a factor of 5 (e.g. $\pi_3/\pi_1 = 5$, then the Java version's comparative utility is higher than the C++ version's in the last column of the first table, and all but the second column of the second table. Another user might consider the danger of a Java-based simulator being unexpectedly overcome by garbage-collection overhead, and weight the functional utility of the C++ based simulators to be twice that of the Java based simulators.

7 Conclusions

This paper considers the problem of comparing network simulators that have different execution speed, memory demand, and features. Our approach recognizes that only a user can balance his interests and requirements among these characteristics. We explore the area from the point of view of utility functions. Our primary interest is in identifying characteristics that utility functions may have that tend to favor a faster simulator over a slower one, or vice-versa.

A common thread among our results is that for the slower simulator to achieve larger utility, its functional utility must be sufficiently larger than that of the faster simulator. Another common thread is that utility functions whose structure don't encourage simulation of maximal model sizes tend to favor the faster simulator. Other results show how much longer the slower simulator will have to run to achieve a utility as large as a faster simulator, and identify conditions under which optimal utility is achieved running the smallest model size of interest, or is achieved running the largest model size possible. Some results make formal statements about the tradeoff between speed and model size necessary for the slower simulator to achieve highest utilization.

We then examined four different simulators from the point of view of utility theory, using two different utility functions, with a variety of parameters. We see that the parameters definitely matter. The fastest simulator sometimes achieves the highest utility, but more often (on the examples considered) a simulator that is 25% slower but simulates models 4 times as large achieves the highest utility.

References

- [1] F. Bacchus and A.J. 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.E. Bell, H. Raiffa, and A. Tversky (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.M. Gorman. The structure of utility functions. *Review of Economic Studies*, 35:367–390, 1968.
- [5] <http://www.cs.dartmouth.edu/~ghyan/dassfnet/overview.htm>.
- [6] <http://www.isi.edu/nsnam/ns/>.
- [7] R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, New York, 1976.
- [8] David Nicol. Scalability of network simulators revisited. Submitted for publication. Pre-publication version at <http://www.cs.dartmouth.edu/~nicol/papers/revisited.pdf>.
- [9] Hung-Ying Tyan and Jennifer Hou. Design, realization, and evaluation of a component-based compositional network simulation environment. In *Proceedings of the 2002 Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Antonio, Texas, January 2002. Society for Computer Simulation.
- [10] www.javasim.org.
- [11] www.javasim.org/comparison.html.
- [12] www.ssfnet.org.