Static Strategies for Worksharing with Unrecoverable Interruptions

Anne Benoit, *Member, IEEE,* Yves Robert, *Fellow, IEEE,* Arnold L. Rosenberg, *Fellow, IEEE,* and Frédéric Vivien, *Member, IEEE*

Abstract—One has a large workload that is "divisible"—its constituent work's granularity can be adjusted arbitrarily—and one has access to p remote computers that can assist in computing the workload. How can one best utilize these computers? Complicating this question is the fact that each remote computer is subject to interruptions (of known likelihood) that kill all work in progress on it. One wishes to orchestrate sharing the workload with the remote computers in a way that maximizes the expected amount of work completed. Strategies are presented for achieving this goal, by balancing the desire to checkpoint often—thereby decreasing the amount of vulnerable work at any point—vs. the desire to avoid the context-switching required to checkpoint. Schedules must also temper the desire to replicate work, because such replication diminishes the effective remote workforce. The current study demonstrates the accessibility of strategies that provably maximize the expected amount of work when there is only one remote computer (the case $p = 1$) and, at least in an asymptotic sense, when there are two remote computers (the case $p = 2$); but the study strongly suggests the intractability of exact maximization for $p \geq 2$ computers, as work replication on multiple remote computers joins checkpointing as a vehicle for decreasing the impact of work-killing interruptions. We respond to that challenge by developing efficient heuristics that employ both checkpointing and work replication as mechanisms for decreasing the impact of work-killing interruptions. The quality of these heuristics, in expected amount of work completed, is assessed through exhaustive simulations that use both idealized models and actual trace data.

Index Terms—Fault-tolerance, Fault-aware scheduling, Divisible load, Risk function, Probabilities

1 Introduction

Technological advances and economic constraints have engendered a variety of modern computing platforms that allow a person who has a massive, compute-intensive workload to enlist the help of others' computers in executing the workload. The resulting cooperating computers may belong to a nearby or remote cluster (of "workstations"; cf. [2]), or they could be geographically dispersed computers that are available under one of the increasingly many modalities of Internet-based computing—such as Grid computing (cf. [3], [4], [5]), global computing (cf. [6]), or volunteer computing (cf. [7]). In order to avoid unintended connotations concerning the organization of the remote computers, we avoid evocative terms such as "cluster" or "grid" in favor of the generic "assemblage."

Advances in computing power never come without cost. The new "collaborative" platforms add various types of *uncertainty* to the list of concerns that must be addressed as one prepares a computational workload for allocation to the available computers. When one allocates work over the Internet, for instance, one must prepare for computers to produce results much more slowly than anticipated, even, possibly, failing ever to complete their allocated work. The current paper follows in the footsteps of sources such as [8], [9], [10], [11], [12], [13], which present analytic studies of algorithmic techniques for coping with uncertainty in computational settings. Whereas most of these sources address the uncertainty of the computers in an assemblage one computer at a time, we view the assemblage here as a "team" wherein one computer's shortcomings can be compensated for by other computers, most notably if we judiciously *replicate work*, i.e., allocate some work to more than one computer.

The problem we study. We have a large computational workload whose constituent work is *divisible*, in the sense that one can subdivide chunks of work into arbitrary granularities (cf. [14]). We also have access to $p \geq 1$ identical "remote" computers to help us compute the workload via *worksharing*, a modality of collaborative computing in which the owner of the workload allocates work to remote computers that are available; cf. [15]. In the current paper we only study *homogeneous* assemblages of remote computers, in order to concentrate only on the problem of coping with uncertainty within an assemblage. We hope to focus in later work on the added complexity of coping with uncertainty within a *heterogeneous* assemblage.

We address here the most draconian type of uncertainty that can plague an assemblage of computers, namely, vulnerability to *unrecoverable interruptions* that kill all work currently in progress on the interrupted computer. We strive to cope with such interruptions—whether they arise from hardware failures or from a loaned/rented computer's being reclaimed by its owner, as during an episode of *cycle-stealing* (cf. [8], [10], [13],

A portion of this research appeared at the 23rd IEEE International Parallel and Distributed Processing Symposium [1].

- *A. Benoit, Y. Robert, and F. Vivien are with the École normale supérieure de Lyon and the Université de Lyon, France.*
- *A. Benoit and Y. Robert are with the Institut Universitaire de France. Their research was supported in part by the ANR StochaGrid project.*
- *A. Rosenberg is with Colorado State University, USA. His research was supported in part by US NSF Grants CNS-0842578 and CNS-0905399.*
- *F. Vivien is with INRIA, France.*

[16], [17]). The two tools that we employ to cope with these interruptions are *checkpointing*, which saves already completed work so that it will not be killed by an interruption, and *work replication*, which allocates some work to more than one remote computer. The only extrinsic resource to help us use these tools judiciously is our assumed *a priori* knowledge of the risk of a computer's having been interrupted—which we assume is the same for all computers.[1]

The goal of our study. We strive to maximize the *expected amount of work that gets computed by the assemblage of computers*, no matter which, or how many, computers get interrupted. We call a schedule that achieves this goal *optimal*. Thus, we are positing that, within our application, even partial output is meaningful. The annotation of metagenomics data is a timely such application: one has a large number of DNA fragments to classify (as *eukaryotes, prokaryotes*, etc.). One ideally wants to have all the DNA fragments processed, but the result of the classification is meaningful even if the annotation is fragmentary—which just artificially augments the "unknown" category.

The challenges. The challenges of scheduling a workload on interruptible remote computers can be described in terms of two dilemmas. *(1)* Sending work to remote computers in small chunks lessens vulnerability to interruption-induced losses, but it also increases the impact of per-work-unit overhead and minimizes the opportunities for "parallelism" within the assemblage of remote computers. *(2)* Replicating work lessens our vulnerability to interruption-induced losses, but it also minimizes the expected productivity advantage from having access to many remote computers. (The pros and cons of work replication are discussed at length in [18].)

Approaches to the challenges. *(1) "Chunking" our workload*. We strive to balance overhead costs with the risk of interruptions, thereby coping with the first dilemma, by allocating work to each remote computer as a sequence of carefully sized *chunks*.[2] This approach, which is advocated in [10], [13], [16], [17], allows each remote computer to *checkpoint* according to some schedule, thereby protecting its work from the threat of work-killing interruptions. *(2) Replicating work*. We strive to give all chunks of work a high likelihood of being computed successfully, despite the second dilemma, by using the (known) probability of interruptions to choose when and where to allocate some chunks of work to more than one remote computer.

Because the costs of checkpointing and of communicating with remote computers are typically high in time and overhead, we limit such activities by orchestrating work allocation (and replication) in an *a priori*, static manner, rather than dynamically, in response to observed interruptions. While we thereby duplicate work unnecessarily when there are few interruptions among the remote computers, we thereby prevent *our* computer, which is the server in the studied scenario, from becoming a bottleneck when there are many interruptions.

Summary. We assume: that we know the instantaneous probability that a remote computer will have been interrupted by time t; that this probability is the same for all remote computers and that it increases with the amount of time that a computer has been available (whether working or not). These assumptions, which we share with [10], [13], [16], seem to be necessary in order to derive scheduling strategies that *provably* maximize the expected amount of work completed. As noted earlier, the challenge of allowing individual computers to differ in power or to have different probabilities of being interrupted must await a sequel to the current study.

Main results. Our early work on this topic is reported somewhat sketchily in the conference report [1]. The current paper is dedicated to fleshing out and extending the (analytical) portion of that report that deals with one and two remote computers; we also expand in an analogous way on the portion of [1] that deals (experimentally) with arbitrary numbers of remote computers, and we extend our heuristics to handle *arbitrary* interruption-risk models. Our results that deal with one remote computer appear in Section 3. We describe the one-computer schedules from [1] that are *exactly* optimal in work production for the interruption-risk model in which risk increases *linearly* with time, both for scenarios that assess per-chunk overheads and those that do not. We end Section 3 with a schedule that is *asymptotically* optimal in work production for one remote computer under *arbitrary* interruption-risk models. The main focus of the current paper expands on and extends the study in [1] regarding worksharing with two remote computers. The difficulty of this extension forces us to focus only on scenarios that do *not* assess per-chunk overheads—what we call the *free-initiation model*; however, one can convert our free-initiation results to results for scenarios that *do* assess per-chunk overheads, that are predictably close to optimal in work production (see Theorem 1 in Section 2.2.4). The difficulty of the two-computer case also forces us, in certain situations, to settle for schedules that are *asymptotically* optimal. Our main results provide the following insights into the worksharing problem for two remote computers.

- *Scheduling guidelines*. Theorem 5 provides guidelines for crafting optimal schedules for any worksharing scenario in which the risk of a computer's being interrupted never decreases with time.
- *Asymptotically optimal schedule, under arbitrary risk*. Theorem 6 provides, for each integer $n > 0$, a two-computer schedule $\Sigma(n)$ that deploys the workload in n chunks. Under any interruption-risk model, the expected work production of the schedules $\Sigma(n)$

1. As in [10], [13], [16], our scheduling strategies can be adapted to use statistical, rather than exact, knowledge of the risk of interruption—albeit at the cost of weakened performance guarantees.

2. We use the generic "chunk" instead of "task" to emphasize tasks' divisibility (which precludes *atomic* tasks).

tends to the optimal limit as n grows without bound. (We term this *asymptotic optimality*.)

- *Exactly optimal schedule, under linear risk, with single-chunk deployment*. Theorem 7 describes a schedule that deploys work in a single chunk to the two remote computers, in a "symmetric" manner (in a sense explained in the theorem). When the risk of interruption grows linearly with time, the schedule is exactly optimal among symmetric schedules.
- *Asymptotically optimal schedule, under linear risk, with multi-chunk deployment*. Algorithm 1 produces, for any given $n > 0$, n-chunk worksharing schedules that are analyzed in Theorem 8. Depending on the size of the workload, the algorithm's schedules are either exactly or asymptotically optimal when the risk of interruption grows linearly with time.

After the study of the two-computer case, we turn to two complementary studies of simple, well-structured heuristics for sharing work with arbitrary numbers of remote computers that are subject to unrecoverable interruptions. Section 5 is devoted to developing strategies that allow one to craft such heuristics. The strategies are instantiated in six heuristics that produce schedules whose structures derive from differing intuitions about how to complete a lot of work in expectation, despite the risk of interruption. The schedules produced by the *greedy* heuristic are found to dominate the other heuristics' schedules in expected work production, albeit at the cost of a bit more computation. The second phase of the current study (Section 6) extends our study of the linear risk function by comparing the schedules produced by the best of the preceding six heuristics against those produced by four simple heuristics. We determine via simulations which heuristics perform better on workloads of varying sizes, varying numbers of remote computers, varying checkpointing granularity, varying checkpointing overheads, and varying degrees of work replication. As in Section 5, the so-called *greedy* heuristic is found to dominate the others. Section 7 takes our study beyond the linear risk model by adapting our new heuristics for use with arbitrary risk functions. The adapted heuristics are then evaluated using actual traces. Finally we conclude in Section 8.

Related work. The literature contains relatively few rigorously analyzed scheduling algorithms for interruptible "parallel" computing in assemblages of computers. Among those we know of, only [8], [10], [13], [16], [17] deal with an *adversarial* model of interruptible computing. One finds in [8] a randomized scheduling strategy which, with high probability, completes within a logarithmic factor of the optimal fraction of the initial workload. In [10], [13], [16], [17], the scheduling problem is viewed as a game against a malicious adversary who seeks to interrupt each remote computer in order to kill all work in progress. (Also, [1] is a preliminary version of the current paper.) Among the experimental sources, [19] studies the use of task replication on a heterogeneous desktop grid whose constituent computers may become definitively unavailable; the objective is to eventually process all work. In a similar context, [20] aims at minimizing both the completion time of applications and the amount of resources used.

There is a large literature on scheduling divisible workloads on assemblages of computers that are not vulnerable to interruptions. We refer the reader to [14] and its myriad intellectual progeny; another good start is [21]—a thorough study of divisible load scheduling on star and tree networks—or [22]. Also on the subject of divisibl workloads, it is shown in [22], [23] how a linear model, such as our free-initiation model, can lead to absurd schedules involving infinitely many infinitely small chunks; we cope with this issue in Section 3.1.1.

We do not enumerate here the many studies of computation on assemblages of remote computers, which focus either on systems that enable such computation or on specific algorithmic applications. However, we point to [24] and [25] as exemplars of the two types of studies.

2 The Technical Framework

We supply the technical details necessary to turn the informal discussion in the Introduction into a framework in which we can develop and rigorously validate scheduling guidelines.

2.1 The Computation and the Computers

We have $W_{(\mathrm{ttl})}$ units of divisible work to execute on an assemblage of $p \geq 1$ identical computers. Each computer is vulnerable to *unrecoverable* interruptions that "kill" all work in progress on it. All computers share the same instantaneous probability of being interrupted, which we know exactly. This probability increases with the amount of time the computer has been operating—whether it has been computing or not.

As discussed in the Introduction, the danger of losing work in progress when an interruption incurs mandates that we not just divide our workload into $W_{(\mathrm{ttl})}/p$ equal-size allocations and deploy one chunk on each computer in the assemblage. Instead, we "protect" our workload as best we can, by:

- partitioning it into *chunks* of possibly different sizes; "chunk" is our term for a unit of work that we allocate to a computer;
- prescribing a schedule for allocating chunks to computers;
- allocating some chunks to more than one computer, as a divisible-load analogue of work replication.

As noted in the Introduction, we treat intercomputer communication as a resource to be used very sparingly. Specifically, in order to avoid having *our* computer become a communication bottleneck, we orchestrate chunk replication in an *a priori*, static manner—rather than dynamically, in response to observed interruptions. This conservative strategy may duplicate work unnecessarily when there are few or no interruptions.

2.2 Modeling Interruptions and Expected Work

2.2.1 The interruption model

Within our study, all computers share the same *risk function*, i.e., the same instantaneous probability, $Pr(w)$, of having been interrupted by the end of "the first w time units." We measure time via work units that *could be* completed "successfully" if there is no interruption. In other words, "the first w time units" is the period of time that a computer would need to complete w units of work *if* it started working on them when the entire worksharing episode begins, *and* it succeeded in completing them. This time scale, which is shared by all computers, thus uses the start of the worksharing episode as the baseline moment for measuring time and risk. Recall that $Pr(w)$, which we assume that we know *exactly*, cannot decrease as w increases.

It is useful to generalize the preceding measure of risk by allowing many baseline moments. We denote by $Pr(s, w)$ the probability that a computer *has not been* interrupted during the first s "time units" but *has been* interrupted by "time" $s + w$. Thus, $Pr(w) = Pr(0, w)$, and $Pr(s, w) = Pr(s+w) - Pr(s)$. We let[3] $\kappa \in (0, 1]$ be a constant that weights our probabilities. We illustrate the role of κ as we introduce the risk function which is our main focus in the current study.

Linearly increasing risk. The risk function that is the main focus of our study is the *linear risk function* $Pr(w) = \kappa w$. It is the most natural model of risk in the absence of further information: the risk of a remote computer's being interrupted grows linearly with the time that the computer has been available, or equivalently, to the amount of work it could have done. The relevant probability density function is then $\mathrm{d}Pr = \kappa \mathrm{d}t$ when $t \in [0, 1/\kappa]$ and 0 otherwise, so that

$$Pr(s, w) \;=\; \min\left\{1,\; \int_{s}^{s+w} \kappa \mathrm{d}t\right\} \;=\; \min\{1, \kappa w\}.$$

The constant $1/\kappa$ recurs often in our analyses, since it can be viewed as the time by which an interruption is certain to have occurred, i.e., will have occurred with probability 1. To enhance legibility of the rather complicated expressions that populate our analyses, we denote the quantity $1/\kappa$ by X.

We point out that the linear risk model is very relevant to cycle-stealing episodes. Consider the following scenario: on Friday evening, a PhD student has a large set of simulations to run. S/he has access to a set of computers from the lab, but each computer can be reclaimed at any instant by its owner. In any case, everybody will be back to work on Monday 8am. What is the student's best strategy? How much simulation data should s/he send to, and execute on, each available computer?

3. As usual, we use parentheses to denote *open* boundaries for real intervals and brackets to denote *closed* boundaries.

2.2.2 Expected work production

We use risk functions to help us find an efficient way to chunk work for, and allocate work to, the remote computers, with the goal of maximizing the expected amount of work completed by the assemblage. Let $\mathsf{W}_{(\mathrm{cmp})}$ be the random variable whose value is the number of work units that the assemblage completes successfully under a given scheduling regimen. Our goal is to maximize the expected value of $\mathsf{W}_{(\mathrm{cmp})}$.

We perform our study using two cost models as we contemplate how to schedule large workloads. The models differ in how they assess chunk execution-times as functions of chunk sizes, in the light of the two classes of costs incurred during each communication or checkpointing, *viz.* the costs that are proportional to the size of a chunk and those that are fixed constants. When chunks are very large, the fixed costs are negligible compared to the size-proportional ones. This suggests that one can safely ignore the fixed costs. But one must be careful! If one ignores the fixed costs, then there is never a disincentive to deploying the workload in[4] $n+1$ chunks rather than n; in fact, this increases the expected work production! However, increasing the number of chunks tends to make chunks smaller—which increases the significance of the fixed costs! We deal with this dilemma by recognizing two cost models and striving for optimal schedules under each.

1) The *free-initiation model* accounts for only the size-proportional costs of worksharing. Because it focuses on situations wherein the fixed costs are negligible compared to the size-proportional ones, it does *not* assess a per-chunk fixed cost.
2) The *charged-initiation model* accounts for both the fixed and the size-proportional costs of worksharing. It thus reflects in greater detail the costs incurred by real computing systems.

The *free-initiation* model. Our results under this model approximate reality well when we allocate work in large chunks. This occurs, for instance, when large fixed costs for each communication or checkpointing event lead us to have each remote computer do a substantial amount of work between successive events, in order to amortize these costs. In such situations, we keep chunks large by placing a bound on the number of scheduling "rounds," in order to counteract this model's tendency to increase the number of "rounds" indefinitely. Importantly also: knowing the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under the free-initiation model affords us easy bounds on the corresponding expected value under the charged-initiation model (see Theorem 1 in Section 2.2.4). Such bounds are quite valuable whenever the charged-initiation model is prohibitively difficult to analyze directly.

Within the free-initiation model, we denote the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under a given schedule Σ, with workload $W_{(\mathrm{ttl})}$, by $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma)$, the superscript "f"

4. Throughout, n denotes a positive integer.

denoting "free"(-initiation). The value of this expectation is

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) \;=\; \int_0^\infty Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u \text{ under } \Sigma)\; du.$$

The *charged-initiation* model. This model is much harder to analyze than the free-initiation model, even when there is only one remote computer. In compensation, this model often allows one to determine analytically the best numbers of chunks and of "rounds," even when there are *multiple* remote computers. Under this model, the overhead for each additional chunk is a fixed cost, ε work units, that is added to the cost of computing each chunk. ε could represent, e.g., the setup time for a communication or the overhead of a checkpoint.

We denote by $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma)$ the expected value, under this model, of $\mathsf{W}_{(\mathrm{cmp})}$, for a given regimen Σ and workload $W_{(\mathrm{ttl})}$; the superscript "c" denotes "charged" (-initiation). This expectation is

$$E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma) \;=\; \int_0^\infty Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u + \varepsilon)\; du.$$

2.2.3 Observing the model on one remote computer

In this section, we consider the single computer case both for the free-initiation model and the charged-initiation one, and explain how to optimize the expected work-production.

The *free-initiation* model. We calculate $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma)$ for an arbitrary risk function Pr, for three cases wherein Σ deploys the entire workload on a single computer. To enhance legibility, let the phrase "under Σ" within the expression "$Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u$ under $\Sigma)$" be specified implicitly by context.

Deployment in one chunk. Regimen Σ_1 allocates the entire workload $W_{(\mathrm{ttl})}$ in a single chunk; we have

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1) \;=\; W_{(\mathrm{ttl})}\left(1 - Pr(W_{(\mathrm{ttl})})\right). \qquad (1)$$

Deployment in two chunks. Regimen Σ_2 partitions the workload into two chunks of respective sizes $\omega_1 > 0$ and $\omega_2 > 0$, where $\omega_1 + \omega_2 = W_{(\mathrm{ttl})}$:

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2) \;&=\; \textstyle\int_0^{\omega_1} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \\ &\quad + \textstyle\int_{\omega_1}^{\omega_1+\omega_2} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \\ &= \omega_1(1 - Pr(\omega_1)) + \omega_2(1 - Pr(\omega_1 + \omega_2)). \end{aligned}$$

Deployment in n chunks. Regimen Σ_n partitions the workload into n chunks of respective sizes $\omega_1 > 0$, $\omega_2 > 0$, …, $\omega_n > 0$, where $\omega_1 + \cdots + \omega_n = W_{(\mathrm{ttl})}$:

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n) \;&=\; \int_0^{\omega_1} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \;+\cdots \\ &\quad + \int_{\omega_1+\cdots+\omega_{n-1}}^{\omega_1+\cdots+\omega_{n-1}+\omega_n} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \\ &= \omega_1(1 - Pr(\omega_1)) \;+\cdots \qquad (2) \\ &\quad + \omega_n(1 - Pr(\omega_1 + \cdots + \omega_n)). \end{aligned}$$

The *charged-initiation* model. Mimicking the development for the free-initiation model: When the entire workload is deployed as a single chunk, we have $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma_1) = W_{(\mathrm{ttl})}\left(1 - Pr(W_{(\mathrm{ttl})} + \varepsilon)\right)$, and when it is deployed as two chunks of respective sizes ω_1 and ω_2, we have $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma_2) = \omega_1(1 - Pr(\omega_1 + \varepsilon)) + \omega_2(1 - Pr(\omega_1 + \omega_2 + 2\varepsilon))$. In general, $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma_n)$ is the charged-initiation analogue of the free-initiation expectation $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$.

Optimizing expected work-production. The ultimate goal of our study is to learn how to craft schedules Σ that maximize $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma)$ (or $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma)$). As a first step toward crafting optimal schedules, we observe that under many risk functions—including the linear risk function—the remote computers are *certain* to have been interrupted no later than a known eventual time. In such situations, one might get more work done, in expectation, by not deploying the entire workload: one could increase the expectation by making the last deployed chunk even a tiny bit smaller than needed to deploy all $W_{(\mathrm{ttl})}$ units of work. (We observe this in Theorem 2 for the free-initiation model and in Theorem 4 for the charged-initiation model.) Thus, when scheduling a single remote computer,

- select n chunk sizes that sum to *at most* $W_{(\mathrm{ttl})}$,
- select from our workload n chunks having these respective sizes,
- schedule the deployment of these chunks

in a way that maximizes the expected amount of work that is completed. We formalize this goal for the free-initiation model via the function $E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})}) = \max\{\omega_1(1 - Pr(\omega_1)) + \cdots + \omega_n(1 - Pr(\omega_1 + \cdots + \omega_n))\}$, where the maximization is over all n-tuples $\{\omega_1 \geq 0,\ \ldots,\ \omega_n \geq 0\}$ such that $\omega_1 + \cdots + \omega_n \leq W_{(\mathrm{ttl})}$.

2.2.4 Relating the models

One can bound the expected work completed under the charged-initiation model via the analogous quantity for the free-initiation model—which is one justification for our focus on the simpler model. The reader can verify the following for the case with one remote computer ($p = 1$) by direct calculation from (2) and its charged-initiation analogue. The theorem remains valid in the general case (with several computers).

Theorem 1: (Relating the models) Let $E_n^{(\mathrm{c})}(W_{(\mathrm{ttl})})$ (resp., $E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})})$) denote the optimal n-chunk expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under the *charged*-initiation model (resp., the *free*-initiation model) under an arbitrary risk function Pr. For any collection of p remote computers,

$$E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})}) \geq E_n^{(\mathrm{c})}(W_{(\mathrm{ttl})}) \geq E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})}) - n\varepsilon. \qquad (3)$$

Proof: The lefthand inequality in (3) being obvious, we focus just on the righthand inequality, which turns out to be subtler than one might expect.

Let Σ be an optimal schedule for p remote computers and risk function Pr under the free-initiation model. Σ deploys the workload $W_{(\mathrm{dpl})}$ in n chunks, $\mathcal{W}_1$, …, $\mathcal{W}_n$,

of respective sizes $\omega_1 > 0, \ldots, \omega_n > 0$. For $j \in [1,p]$, $\Sigma(j,k)$ denotes the kth chunk executed on computer j under Σ. In other words, computer j executes chunks in the order $\Sigma(j,1), \Sigma(j,2), \ldots, \Sigma(j,n)$.[5]

Note that distinct chunks may overlap; i.e., we may have $\mathcal{W}_i \cap \mathcal{W}_j \neq \emptyset$ even when $i \neq j$. We therefore *partition* $W_{(\mathrm{dpl})}$ into $\{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ so that, for any $\mathcal{X}_i$ and $\mathcal{W}_j$, either $[\mathcal{X}_i \subseteq \mathcal{W}_j]$ or $[\mathcal{X}_i \cap \mathcal{W}_j = \emptyset]$. Thus, for all $i \in [1,m]$, if Σ schedules any part of $\mathcal{X}_i$ on computer j, then the latter attempts to compute $\mathcal{X}_i$ in its entirety. Let Π_i be the set of computers that $\mathcal{X}_i$ is scheduled on. For $j \in \Pi_i$, $\sigma(j,i)$ is the rank of the first chunk scheduled on computer j that contains $\mathcal{X}_i$; formally $\sigma(j,i) = \min\{k | \mathcal{X}_i \subset \Sigma(j,k)\}$.

Define schedule Σ' from Σ as follows. For each i in $[1,n]$, let $\omega_i' = \max\{0, \omega_i - \varepsilon\}$, and let $\mathcal{W}_i'$ be any size-ω_i' subset of $\mathcal{W}_i$. Σ' deploys chunks $\mathcal{W}_1', \ldots, \mathcal{W}_n'$ in the *exact* manner that Σ deploys chunks $\mathcal{W}_1, \ldots, \mathcal{W}_n$ on those same computers, except that null chunks are skipped (to avoid the cost ε). We account for these zero-length chunks in the following equations, via the function

$$\mathbf{1}_{\omega_\mathbf{i}'} \;=\; \begin{cases} 1 & \text{if } \omega_i' \neq 0 \\ 0 & \text{if } \omega_i' = 0. \end{cases}$$

We then define $\mathcal{X}_i'$, Π_i' and $\sigma'(j,i)$ as we did $\mathcal{X}_i$, Π_i and $\sigma(j,i)$, except that we now insist that each $\mathcal{X}_i'$ be a subset of some $\mathcal{X}_k$; we thus refine the partition $\mathcal{X}$ to the partition $\mathcal{X}'$. Let $\mathcal{X}_{\tau(i)}$ be the element of $\mathcal{X}$ that contains $\mathcal{X}_i'$. Finally, let $\mathcal{I}'$ be the largest subset of $\mathcal{X}'$ such that:

$$\forall i \in \mathcal{I}',\ \{j \mid \mathcal{X}_i' \subset \mathcal{W}_j'\} \;=\; \{j \mid \mathcal{X}_{\tau(i)} \subset \mathcal{W}_j\}.$$

If $\mathcal{X}_i'$ does not belong to $\mathcal{I}'$, there exists a chunk $\mathcal{W}_j$ such that some piece of work in $\mathcal{X}_{\tau(i)}$ belongs to $\mathcal{W}_j$ but not to $\mathcal{W}_j'$: $\mathcal{X}_i' \subset \mathcal{W}_j \setminus \mathcal{W}_j'$, $\cup_{i \notin \mathcal{I}'} \mathcal{X}_i' \subset \cup_{i=1}^n \mathcal{W}_i \setminus \mathcal{W}_j'$, and

$$\sum_{i \notin \mathcal{I}'} |\mathcal{X}_i'| = \left| \bigcup_{i \notin \mathcal{I}'} \mathcal{X}_i' \right| \leq \left| \bigcup_{i=1}^n \mathcal{W}_i \setminus \mathcal{W}_j' \right| \leq \sum_{i=1}^n |\mathcal{W}_i \setminus \mathcal{W}_j'| \leq n\varepsilon.$$

Since Σ' implicitly specifies a ($\leq n$)-chunk schedule under the charged-initiation model, its expected work production cannot exceed that of the best ($\leq n$)-chunk schedule regimen under the charged-initiation model:

$$E_n^{(\mathrm{c})}(W_{(\mathrm{ttl})}) \geq E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma').$$

5. Clearly, having each computer attempt all n chunks cannot decrease the overall expectation.

However,

$$\begin{aligned}
&E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma') \\
&= \sum_{i=1}^{m'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_i'} Pr\left(\sum_{k=1}^{\sigma'(j,i)} (\omega'_{\Sigma(j,k)} + \varepsilon \mathbf{1}_{\omega'_{\mathbf{\Sigma(j,k)}}}) \right) \right) \\
&= \sum_{i=1}^{m'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_i'} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \mathbf{1}_{\omega'_{\mathbf{\Sigma(j,k)}}} (\omega'_{\mathbf{\Sigma(j,k)}} + \varepsilon) \right) \right) \\
&= \sum_{i=1}^{m'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_i'} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \mathbf{1}_{\omega'_{\mathbf{\Sigma(j,k)}}} \omega_{\mathbf{\Sigma(j,k)}} \right) \right) \\
&\geq \sum_{i=1}^{m'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_i'} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \omega_{\Sigma(j,k)} \right) \right) \\
&\geq \sum_{i \in \mathcal{I}'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_i'} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \omega_{\Sigma(j,k)} \right) \right) \\
&= \sum_{i \in \mathcal{I}'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_{\tau(i)}} Pr\left(\sum_{k=1}^{\sigma(j,\tau(i))} \omega_{\Sigma(j,k)} \right) \right) \\
&= E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) \\
&\quad - \sum_{i \notin \mathcal{I}'} |\mathcal{X}_i'| \left(1 - \prod_{j \in \Pi_{\tau(i)}} Pr\left(\sum_{k=1}^{\sigma(j,\tau(i))} \omega_{\Sigma(j,k)} \right) \right) \\
&\geq E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) - \sum_{i \notin \mathcal{I}'} |\mathcal{X}_i'| \\
&= E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) - \left| \bigcup_{i \notin \mathcal{I}'} \mathcal{X}_i' \right| \\
&\geq E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})}) - n\varepsilon
\end{aligned}$$

which yields the righthand bound. □

3 Scheduling One Remote Computer

This section derives optimal or asymptotically schedules when there is only a single remote computer.

3.1 Scheduling under the Free-Initiation Model

Under the free-initiation model, we are able to derive rather strong results: an exactly optimal worksharing schedule under the linear risk function (Section 3.1.1) and an asymptotically optimal one for arbitrary risk functions (Section 3.1.2).

3.1.1 An optimal schedule for the linear risk model

Even this simplest of the scenarios we study has complicating subtleties. The risk of losing work because of an interruption molds our scheduling strategies, even when there is only one remote computer and when additional chunks of work—which betoken additional communications and/or checkpoints—incur no cost. When we instantiate the expectations derived in Section 2.2.2 with the linear risk function, we discover the following. When $W_{(\mathrm{ttl})} \leq 1/\kappa$, the expected amount of work achieved when one deploys the entire workload as a *single chunk* is $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1) = W_{(\mathrm{ttl})} - \kappa(W_{(\mathrm{ttl})})^2$. The analogous

quantity when one deploys the workload as *two chunks*, of respective sizes $\omega_1 > 0$ and $\omega_2 > 0$, with $\omega_1 + \omega_2 = W_{(\mathrm{ttl})}$, is $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2) = W_{(\mathrm{ttl})} - W_{(\mathrm{ttl})}^2\kappa + \omega_1\omega_2\kappa$. It follows that

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2) - E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1) \;=\; \omega_1\omega_2\kappa \;>\; 0.$$

Thus, for any fixed total workload, one increases the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ by deploying the workload as two chunks rather than one—*no matter how one sizes the chunks*. In fact, this trend continues indefinitely: the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ for the optimal schedule strictly increases with the number of chunks allowed (Theorem 2). This fact identifies a weakness of the free-initiation model: the (unattainable) "optimal" strategy would deploy infinitely many infinitely small chunks. We formalize and flesh out this observation.

Theorem 2 ([1]): (Optimal schedule: free initiation, linear risk) One wishes to deploy $W_{(\mathrm{ttl})}$ units of work to a single computer in at most n chunks, for some $n > 0$.
The goal. To maximize the expectation of $\mathsf{W}_{(\mathrm{cmp})}$.
The unique optimal regimen. We deploy only $W_{(\mathrm{dpl})} = \min\left\{W_{(\mathrm{ttl})},\ \frac{n}{n+1}X\right\}$ units of work,[6] and we let each chunk comprise $W_{(\mathrm{dpl})}/n$ units of work.
In expectation, this regimen completes $E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})}) = W_{(\mathrm{dpl})} - \frac{1}{2}(1 + 1/n)W_{(\mathrm{dpl})}^2\kappa$ units of work.

Proof: Let us partition the $W_{(\mathrm{ttl})}$-unit workload into $n+1$ chunks, of respective sizes $\omega_1 \geq 0$, ..., $\omega_n \geq 0$, $\omega_{n+1} \geq 0$, with the intention of deploying the first n of these chunks.

> (a) Our assigning the first n chunks *nonnegative*, rather than *positive* sizes affords us a convenient way to talk about "at most n chunks" using only the single parameter n. (b) By creating $n+1$ chunks rather than n, we allow ourselves to hold back some work in order to avoid what would be a certain interruption of the nth chunk. Formally, exercising this option means making ω_{n+1} positive; declining the option—thereby deploying all $W_{(\mathrm{ttl})}$ units of work—means setting $\omega_{n+1} = 0$.

Each specific such partition specifies an n-chunk schedule Σ_n. Our challenge is to choose the sizes of the $n+1$ chunks in a way that maximizes $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$. To simplify notation, let $Z = \omega_1 + \cdots + \omega_n$ denote the portion of the entire workload that we actually deploy.

Extending the reasoning from the cases $n = 1$ and $n = 2$, one obtains easily from (2) the expression

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n) &= \omega_1(1 - \omega_1\kappa) + \omega_2(1 - (\omega_1 + \omega_2)\kappa) \\ &\quad + \cdots + \omega_n(1 - (\omega_1 + \cdots + \omega_n)\kappa) \\ &= Z - Z^2\kappa + \left[\sum_{1 \leq i < j \leq n} \omega_i\omega_j\right]\kappa. \quad (4) \end{aligned}$$

Standard arguments show that the bracketed sum in (4) is maximized when all ω_i's share the common value Z/n, in which case, the sum achieves the value $\frac{1}{n^2}\binom{n}{2}Z^2\kappa$. Since maximizing the sum also maximizes $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$, simple arithmetic yields:

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n) \;=\; Z \;-\; \frac{n+1}{2n}Z^2\kappa.$$

Viewing this expression for $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$ as a function of Z, we note that the function is unimodal, increasing until $Z = \frac{n}{(n+1)\kappa}$ and decreasing thereafter. Setting this value for Z, gives us the maximum value for $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$, i.e., the value of $E_n^{(\mathrm{f})}(W_{(\mathrm{ttl})})$. The theorem follows. □

Note. Many risk functions, such as the linear risk function, have "built-in" ends to worksharing episodes, because there is an amount of work, call it V, by whose completion, the computer is *certain* to have been interrupted; i.e., the probability of its having been interrupted is 1. For such risk functions, one can improve the equal-chunk schedule $\Sigma(n)$ by choosing chunk sizes based on V instead of on $W_{(\mathrm{ttl})}$. This phenomenon is visible in Theorem 2 and recurs in our study.

3.1.2 Asymptotic optimality under arbitrary risk

The following notation is useful throughout the paper. Say that our workload consists of $W_{(\mathrm{ttl})}$ units of work that we somehow order linearly. We denote by $\langle a, b\rangle$ the sub-workload obtained by eliminating: the initial a units of work and all work beyond the initial b units. For illustration: $\langle 0, W_{(\mathrm{ttl})}\rangle$ denotes the entire workload, $\langle 0, \frac{1}{2}W_{(\mathrm{ttl})}\rangle$ denotes the first half of the workload, and $\langle \frac{1}{2}W_{(\mathrm{ttl})}, W_{(\mathrm{ttl})}\rangle$ denotes the last half of the workload.

We show now that, for one remote computer, a schedule that deploys equal-size chunks is *asymptotically optimal*, no matter what the risk function. Throughout, we employ the phrase "asymptotically optimal" to mean that the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ of the schedule being discussed tends to the expectation of an optimal schedule as n grows without bound.

Theorem 3: (Asymptotically optimal schedule: free initiation, arbitrary risk) One wishes to deploy $W_{(\mathrm{ttl})}$ units of work to a single remote computer in at most n chunks, for some $n > 0$.
The goal. To maximize the expectation of $\mathsf{W}_{(\mathrm{cmp})}$.
An asymptotically optimal schedule, $\Sigma(n)$. One partitions the overall workload into n equal-size chunks $\mathcal{W}_1, \cdots, \mathcal{W}_n$:

$$\text{For each} \quad i \in [1, n]$$
$$\text{Assign:} \quad \left[\mathcal{W}_i \;\leftarrow\; \left\langle \frac{i-1}{n}W_{(\mathrm{ttl})},\ \frac{i}{n}W_{(\mathrm{ttl})}\right\rangle\right]$$

Proof: We denote by $Opt(n)$ an optimal regimen using (at most) n chunks. Under regimen $Opt(n)$, we denote the chunks $\mathcal{W}'_1$, ..., $\mathcal{W}'_n$, and we denote by ω'_i the size of chunk $\mathcal{W}'_i$. Without loss of generality, we can assume that, for any i in $[1..n]$, chunk $\mathcal{W}'_i$ is equal to $\left\langle \sum_{k=1}^{i-1}\omega'_k, \sum_{k=1}^{i}\omega'_k\right\rangle$.

6. Recall that the episode is certain to be interrupted by time X.

Let us consider any strictly positive integer m. We are going to compare the performance of the scheduling regimens $Opt(n)$ and $\Sigma(m)$. For that purpose, we introduce three more notations. First, we denote by α the size of a chunk of $\Sigma(m)$: $\alpha = \frac{W_{(\mathrm{ttl})}}{m}$. Then, for any $i \in [1..m-1]$, let $s(i)$ be the index of the first chunk of $\Sigma(m)$ which starts no sooner than the end of the ith chunk of $Opt(n)$. Formally:

$$s(i) = 1 + \left\lceil \frac{\sum_{k=1}^{i} \omega'_k}{\alpha} \right\rceil.$$

Symmetrically, for any $i \in [1..m]$, let $p(i)$ be the index of the last chunk of $\Sigma(m)$ which ends no later than the beginning of the ith chunk of $Opt(n)$. Formally:

$$p(i) = \left\lfloor \frac{\sum_{k=1}^{i-1} \omega'_k}{\alpha} \right\rfloor.$$

If at least one chunk of $\Sigma(m)$ is fully included in $\mathcal{W}'_i$, $s(i-1)$ is the index of the first such chunk, and $p(i+1)$ the index of the last such chunk.

The overall expectation of $\mathsf{W}_{(\mathrm{cmp})}$ for $Opt(n)$ is:

$$E(W_{(\mathrm{ttl})}, Opt(n)) = \sum_{i=1}^{n} \omega'_i \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right). \quad (5)$$

Let us now consider any chunk $\mathcal{W}'_i$ of $Opt(n)$ (that is, any $i \in [1..n]$). Its contribution to the overall expectation is:

$$e_i = \omega'_i \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right). \quad (6)$$

If $\omega'_i < 2\alpha$, obviously $e_i < 2\alpha$. Otherwise, $\omega'_i \geq 2\alpha$ and there exists at least one chunk of $\Sigma(m)$ which is included in the chunk $\mathcal{W}'_i$. Then, $p(i+1) \geq s(i-1)$ (we extend s by letting $s(0) = 0$). We can then establish:

$$\begin{aligned}
\omega'_i &= \sum_{j=1}^{i} \omega'_j - \sum_{j=1}^{i-1} \omega'_j \\
&= \left(\sum_{j=1}^{i} \omega'_j - p(i+1)\alpha\right) \\
&\quad + (p(i+1)\alpha - (s(i-1)-1)\alpha) \\
&\quad + \left((s(i-1)-1)\alpha - \sum_{j=1}^{i-1} \omega'_j\right) \\
&< \alpha + (p(i+1) - s(i-1) + 1)\,\alpha + \alpha.
\end{aligned}$$

Using this result and Equation (6) we can bound the value of e_i:

$$\begin{aligned}
e_i &< (2\alpha + (p(i+1) - s(i-1) + 1)\,\alpha)\left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right) \\
&\leq 2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right) \\
&\leq 2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha\,(1 - Pr\,(j\alpha)).
\end{aligned}$$

The last inequation holds because $p(i+1)\alpha$ is no greater than $\sum_{j=1}^{i} \omega'_j$ and because Pr is a non decreasing function.

We can now rewrite Equation (5):

$$\begin{aligned}
&E(W_{(\mathrm{ttl})}, Opt(n)) \\
&\quad = \sum_{\substack{1 \leq i \leq n \\ \omega'_i < 2\alpha}} e_i + \sum_{\substack{1 \leq i \leq n \\ \omega'_i \geq 2\alpha}} e_i \\
&\quad < \sum_{\substack{1 \leq i \leq n \\ \omega'_i < 2\alpha}} 2\alpha + \sum_{\substack{1 \leq i \leq n \\ \omega'_i \geq 2\alpha}} \left(2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha\,(1 - Pr\,(j\alpha))\right) \\
&\quad \leq 2n\alpha + \sum_{j=1}^{m} \alpha\,(1 - Pr\,(j\alpha)) \\
&\quad \leq \frac{2n}{m} W_{(\mathrm{ttl})} + E(W_{(\mathrm{ttl})}, \Sigma(m)).
\end{aligned}$$

Therefore, for any positive integers n and m:

$$\begin{aligned}
E(W_{(\mathrm{ttl})}, Opt(n)) - \frac{2n}{m} W_{(\mathrm{ttl})} \\
< \quad E(W_{(\mathrm{ttl})}, \Sigma(m)) \quad \leq \\
E(W_{(\mathrm{ttl})}, Opt(m)). \quad (7)
\end{aligned}$$

$E(W_{(\mathrm{ttl})}, Opt(n))$ is obviously a nondecreasing, upper-bounded (by $W_{(\mathrm{ttl})}$), sequence and it thus converges. By replacing n by $\lfloor \sqrt{m} \rfloor$ in Equation (7), one easily sees that the sequence $E(W_{(\mathrm{ttl})}, \Sigma(m))$ is converging with the same limit.

□

3.2 The Charged-Initiation Model, with Linear Risk

The *charged-initiation* analogue of Theorem 2 is drastically more difficult to deal with. The following result is strikingly similar to an analogous result in [10], despite substantive differences between our model and theirs.

Theorem 4 ([1]): (Optimal schedule: charged initiation, linear risk) One wishes to deploy $W_{(\mathrm{ttl})}$ units of work to a single computer in at most n chunks, for some integer $n > 0$, when $X \geq \varepsilon$.

The goal. To maximize $E_n^{(\mathrm{c})}(W_{(\mathrm{ttl})})$.

The unique optimal schedule. Let

$n_1 = \left\lfloor \frac{1}{2}(1 + \frac{8}{\varepsilon}X)^{\frac{1}{2}} - \frac{1}{2} \right\rfloor$; $n_2 = \left\lfloor \frac{1}{2}(1 + \frac{8}{\varepsilon}W_{(\mathrm{ttl})})^{\frac{1}{2}} + \frac{1}{2} \right\rfloor$.

The unique regimen:

1) deploys the work in $m = \min\{n,\ n_1,\ n_2\}$ chunks;
2) gives the first chunk size
   $\omega_{1,m} = W_{(\mathrm{dpl})}/m + (m-1)\varepsilon/2$
   where
   $$W_{(\mathrm{dpl})} = \min\left\{W_{(\mathrm{ttl})},\ \frac{m}{m+1}X - m\varepsilon/2\right\}. \quad (8)$$
3) inductively, gives the $(i+1)$th chunk size
   $\omega_{i+1,m} = \omega_{i,m} - \varepsilon$.

In expectation, this schedule completes

$$\begin{aligned}
E_n^{(\mathrm{c})}(W_{(\mathrm{ttl})}) = W_{(\mathrm{dpl})} - \frac{m+1}{2m} W_{(\mathrm{dpl})}^2 \kappa \\
- \frac{m+1}{2}\left(W_{(\mathrm{dpl})} - \frac{(m-1)m}{12}\varepsilon\right)\varepsilon\kappa \quad (9)
\end{aligned}$$

units of work.

Note that $E_n^{(\mathrm{c})}(W_{(\mathrm{ttl})})$ is maximal for any value of n not smaller than $\min\{n_1, n_2\}$.

Proof: We proceed by induction on the number n of chunks we want to partition our $W_{(\mathrm{ttl})}$ units of work into. We denote by $\mathcal{E}_{\mathrm{opt}}^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n)$ the maximum expected amount of work that a schedule can complete under such a partition.

Focus first on the case $n = 1$. When work is allocated in a single chunk, the maximum expected amount of total work completed is, by definition:

$$\mathcal{E}_{\mathrm{opt}}^{(\mathrm{c})}(W_{(\mathrm{ttl})}, 1) \;=\; \max_{0 \le \omega_{1,1} \le W_{(\mathrm{ttl})}} \mathcal{E}^{(\mathrm{c})}(\omega_{1,1}) \quad \text{where}$$

$$\mathcal{E}^{(\mathrm{c})}(\omega_{1,1}) \;=\; \max_{0 \le \omega_{1,1} \le W_{(\mathrm{ttl})}} \omega_{1,1}(1 - (\omega_{1,1} + \varepsilon)\kappa).$$

We determine the optimal size of $\omega_{1,1}$ by viewing this quantity as a variable in the closed interval $[0, W_{(\mathrm{ttl})}]$ and maximizing $\mathcal{E}^{(\mathrm{c})}(\omega_{1,1})$ symbolically. We thereby find that $\mathcal{E}^{(\mathrm{c})}(\omega_{1,1})$ is maximized by setting

$$\omega_{1,1} \;=\; \min\left\{W_{(\mathrm{ttl})}, \frac{1}{2\kappa} - \frac{\varepsilon}{2}\right\},$$

so that

$$\mathcal{E}_{\mathrm{opt}}^{(\mathrm{c})}(W_{(\mathrm{ttl})}, 1) = \begin{cases} \dfrac{1}{4\kappa} - \dfrac{\varepsilon}{2} + \dfrac{\varepsilon^2}{4}\kappa \text{ if } W_{(\mathrm{ttl})} > \dfrac{1}{2\kappa} - \dfrac{\varepsilon}{2}, \\ W_{(\mathrm{ttl})} - W_{(\mathrm{ttl})}^2\kappa - W_{(\mathrm{ttl})}\varepsilon\kappa \text{ otherwise.} \end{cases}$$

(Note that $\omega_{1,1}$ has a non-negative size because of the natural hypothesis that $X \ge \varepsilon$.)

We now proceed to general values of n by induction. We begin by assuming that the conclusions of the theorem have been established for the case when the workload is split into $n \ge 1$ positive-size chunks. We also assume that n is no greater than $\min\{n_1, n_2\}$. In other words, we assume that any optimal solution with at most n chunks used n positive-size chunks.

As our first step in analyzing how best to deploy $n+1$ positive-size chunks, we note that the only influence the first n chunks of work have on the probability that the last chunk will be computed successfully is in terms of their cumulative size.

> Let us clarify this last point, which follows from the failure probability model. Denote by A_n the cumulative size of the first n chunks of work in the expectation-maximizing $(n+1)$-chunk scenario; i.e., $A_n = \sum_{i=1}^{n} \omega_{i,n+1}$. Once A_n is specified, the probability that the remote computer will be interrupted while working on the $(n+1)$th chunk depends only on the value of A_n, *not* on the way the A_n units of work have been divided into chunks.

This fact means that once one has specified the cumulative size of the workload that comprises the first n chunks, the best way to partition this workload into chunks is as though it were the only work in the system, i.e., as if there were no $(n+1)$th chunk to be allocated. Thus, one can express $\mathcal{E}_{\mathrm{opt}}(W_{(\mathrm{ttl})}, n+1)$ in terms of A_n (whose value must, of course, be determined) and $\mathcal{E}_{\mathrm{opt}}(A_n, n)$, via the following maximization.

$$\begin{aligned} &\mathcal{E}_{\mathrm{opt}}(W_{(\mathrm{ttl})}, n+1) = \\ &\quad \max\big\{\mathcal{E}_{\mathrm{opt}}(A_n, n) \\ &\qquad + \omega_{n+1,n+1}\left(1 - (A_n + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right)\big\}, \end{aligned}$$

where the maximization is over all values for A_n in which

$$\begin{array}{c} A_n \quad > 0 \\ \text{(allowing for the } n \text{ previous chunks)} \\ \omega_{n+1,n+1} \quad \ge 0 \\ \text{(allowing for an } (n+1)\text{th chunk)} \\ A_n + \omega_{n+1,n+1} \quad \le W_{(\mathrm{ttl})} \\ \text{(because the total workload has size } W_{(\mathrm{ttl})}) \\ A_n + \omega_{n+1,n+1} + (n+1)\varepsilon \;\le\; X \\ \text{(reflecting the risk and cost models)} \end{array}$$

The last of these inequalities acknowledges that the remote computer is certain to be interrupted (with probability 1) before it can complete the $(n+1)$th chunk of work, if its overall workload is no smaller than $X - (n+1)\varepsilon$.

We now have two cases to consider, depending on the size of A_n.

Case 1: $A_n < \dfrac{n}{n+1}X - \dfrac{n}{2}\varepsilon$.

By assumption, the expectation-maximizing regimen deploys A_n units of work via its first n chunks. By induction, expression (9) tells us that the expected amount of work completed by deploying these A_n units is

$$\begin{aligned} &\mathcal{E}_{\mathrm{opt}}^{(c)}(A_n, n) = \\ &A_n - \frac{n+1}{2n}A_n^2\kappa - \frac{n+1}{2}A_n\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa. \end{aligned}$$

Let $W_{(\mathrm{dpl})}$ denote the total work that is actually allocated: $W_{(\mathrm{dpl})} = A_n + \omega_{n+1,n+1}$. In the following calculations, we write $\omega_{n+1,n+1}$ as $W_{(\mathrm{dpl})} - A_n$, in order to represent the $(n+1)$-chunk scenario entirely via quantities that arise in the n-chunk scenario.

We focus on

$$\begin{aligned} &\mathcal{E}^{(1)}(A_n, \omega_{n+1,n+1}) \\ &= \mathcal{E}_{\mathrm{opt}}(A_n, n) + (W_{(\mathrm{dpl})} - A_n)\left(1 - \left(W_{(\mathrm{dpl})} + (n+1)\varepsilon\right)\kappa\right) \\ &= \left(A_n - \tfrac{n+1}{2n}A_n^2\kappa - \tfrac{n+1}{2}A_n\varepsilon\kappa + \tfrac{(n-1)n(n+1)}{24}\varepsilon^2\kappa\right) \\ &\quad + (W_{(\mathrm{dpl})} - A_n)\left(1 - \left(W_{(\mathrm{dpl})} + (n+1)\varepsilon\right)\kappa\right) \\ &= \left(W_{(\mathrm{dpl})} + \tfrac{n+1}{2}\varepsilon\right)A_n\kappa - \tfrac{n+1}{2n}A_n^2\kappa \\ &\quad + W_{(\mathrm{dpl})}(1 - (W_{(\mathrm{dpl})} + (n+1)\varepsilon)\kappa) + \tfrac{(n-1)n(n+1)}{24}\varepsilon^2\kappa. \end{aligned}$$

For a given value of $W_{(\mathrm{dpl})}$, we look for the best value for A_n using the preceding expression. We note first that

$$\frac{\partial \mathcal{E}^{(1)}(A_n, \omega_{n+1,n+1})}{\partial A_n} \;=\; -\frac{n+1}{n}A_n\kappa + W_{(\mathrm{dpl})}\kappa + \frac{n+1}{2}\varepsilon\kappa.$$

We note next that, for fixed $W_{(\mathrm{dpl})}$, the quantity $\mathcal{E}^{(1)}(A_n, \omega_{n+1,n+1})$ begins to increase with A_n and then

decreases. The value for A_n that maximizes this expectation, which we denote $A_n^{(\text{opt})}$, is

$$A_n^{(\text{opt})} \;=\; \min\left\{W_{(\text{ttl})}, \frac{n}{n+1}W_{(\text{dpl})} + \frac{n}{2}\varepsilon\right\}.$$

When $W_{(\text{ttl})} \leq (n/(n+1))W_{(\text{dpl})} + \frac{1}{2}n\varepsilon$, $A_n^{(\text{opt})} = W_{(\text{ttl})}$, meaning that the $(n+1)$th chunk is empty, and the schedule does *not* optimize the expected work. (In the charged-initiation model an empty chunk decreases the overall probability). Consequently, we focus *for the moment* on the case

$$A_n^{(\text{opt})} \;=\; \frac{n}{n+1}W_{(\text{dpl})} + \frac{n}{2}\varepsilon \tag{10}$$

(thereby assuming that $W_{(\text{ttl})} \geq (n/(n+1))W_{(\text{dpl})} + \frac{1}{2}n\varepsilon$). Therefore, we have

$$\begin{aligned}\mathcal{E}^{(1)}(A_n^{(\text{opt})}, \omega_{n+1,n+1}) \;=\; & \\ -\frac{n+2}{2(n+1)}W_{(\text{dpl})}^2\kappa + W_{(\text{dpl})} - \frac{n+2}{2}\varepsilon W_{(\text{dpl})}\kappa & \\ + \frac{n(n+1)(n+2)}{24}\varepsilon^2\kappa.&\end{aligned}$$

We maximize $\mathcal{E}^{(1)}(A_n^{(\text{opt})}, \omega_{n+1,n+1})$ via the preceding expression by viewing the expectation as a function of $W_{(\text{dpl})}$. We discover that $\mathcal{E}^{(1)}(A_n^{(\text{opt})}, \omega_{n+1,n+1})$ is maximized when

$$W_{(\text{dpl})} = W_{(\text{dpl})}^{(\text{opt})} = \min\left\{\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon, W_{(\text{ttl})}\right\}. \tag{11}$$

For this case to be meaningful, the $(n+1)$th chunk must be nonempty, so that $A_n^{(\text{opt})} < Z$; i.e., $Z > \frac{1}{2}n(n+1)\varepsilon$. Therefore, we must simultaneously have:

1) $(n+1)/(n+2)X - \frac{1}{2}(n+1)\varepsilon > \frac{1}{2}n(n+1)\varepsilon$, so that $X > \frac{1}{2}(n+1)(n+2)\varepsilon$, which requires that $n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon} - 3\right)\right\rfloor$.
2) $W_{(\text{ttl})} > \frac{1}{2}n(n+1)\varepsilon$, which requires that $n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\text{ttl})}/\varepsilon} - 1\right)\right\rfloor$.

We can now check the sanity of the result.

$$W_{(\text{dpl})}^{(\text{opt})} + (n+1)\varepsilon \;\leq\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon + (n+1)\varepsilon \;<\; X,$$

because of the just established condition $\frac{1}{2}(n+1)(n+2)\varepsilon < X$. We also have,

$$\begin{aligned}A_n^{(\text{opt})} \;&=\; \tfrac{n}{n+1}W_{(\text{dpl})} + \tfrac{n}{2}\varepsilon \\ &\leq \tfrac{n}{n+1}\left(\tfrac{n+1}{n+2}X - \tfrac{n+1}{2}\varepsilon\right) + \tfrac{n}{2}\varepsilon \\ &= \tfrac{n}{n+2}X \\ &< \tfrac{n}{n+1}X - \tfrac{n}{2}\varepsilon\end{aligned}$$

because $\frac{1}{2}(n+1)(n+2)\varepsilon < X$. Therefore, the solution is consistent with the defining hypothesis for this case—namely, that $A_n < \dfrac{n}{n+1}X - \dfrac{n}{2}\varepsilon$.

Before moving on to case 2, we note that the value (10) does, indeed, extend our inductive hypothesis. To wit, the optimal total amount of allocated work, $W_{(\text{dpl})}^{(\text{opt})}$, has precisely the predicted value, and the sizes of the first n chunks do follow a decreasing arithmetic progression with common difference ε (by using the induction hypothesis). Finally, the last chunk has the claimed size:

$$\omega_{n+1,n+1} \;=\; W_{(\text{dpl})}^{(\text{opt})} - A_n^{(\text{opt})} \;=\; \frac{1}{n+1}W_{(\text{dpl})}^{(\text{opt})} - \frac{n}{2}\varepsilon.$$

We turn now to our remaining chores. We must derive the expectation-maximizing chunk sizes for the second case, wherein A_n is "big." And, we must show that the maximal expected work completion in this second case is always dominated by the solution of the first case—which will lead us to conclude that the regimen of the theorem is, indeed, optimal.

Case 2: $A_n \geq \dfrac{n}{n+1}X - \dfrac{n}{2}\varepsilon$.

By (8), if the current case's restriction on A_n is an *in*equality, then A_n *cannot* be an optimal cumulative n-chunk work allocation. We lose no generality, therefore, by focusing only on the subcase when the defining restriction of A_n is an *equality:*

$$A_n \;=\; \frac{n}{n+1}X - \frac{n}{2}\varepsilon.$$

For this value of A_n, call it $A_n^\star$, we have

$$\begin{aligned}&\mathcal{E}_{\text{opt}}(W_{(\text{ttl})}, n+1) \;= \\ &\quad\max\left(\mathcal{E}_{\text{opt}}(A_n^\star, n)\right. \\ &\qquad\left.+\omega_{n+1,n+1}\left(1 - (A_n^\star + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right)\right),\end{aligned}$$

where the maximization is over all values of $\omega_{n+1,n+1}$ in the closed interval $[0,\ W_{(\text{ttl})} - A_n^\star]$.

To determine a value of $\omega_{n+1,n+1}$ that maximizes $\mathcal{E}_{\text{opt}}(W_{(\text{ttl})}, n+1)$ for $A_n^\star$, we focus on the function

$$\begin{aligned}&\mathcal{E}^{(2)}(A_n, \omega_{n+1,n+1}) \\ &\quad= \mathcal{E}_{\text{opt}}(A_n, n) \\ &\qquad+ \omega_{n+1,n+1}\left(1 - (A_n + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right) \\ &\quad= \left(A_n - \tfrac{n+1}{2n}A_n^2\kappa - \tfrac{n+1}{2}A_n\varepsilon\kappa + \tfrac{(n-1)n(n+1)}{24}\varepsilon^2\kappa\right) \\ &\qquad+ \omega_{n+1,n+1}\left(1 - (A_n + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right) \\ &\quad= -\kappa\omega_{n+1,n+1}^2 + \left(-\tfrac{n+2}{2}\kappa\varepsilon + \tfrac{1}{n+1}\right)\omega_{n+1,n+1} \\ &\qquad+ \tfrac{n\left((n+1)^2(n+2)\varepsilon^2\kappa^2 - 12(n+1)\varepsilon\kappa + 12\right)}{24(n+1)\kappa}.\end{aligned}$$

Easily,

$$\frac{\partial\mathcal{E}^{(2)}(A_n, \omega_{n+1,n+1})}{\partial\omega_{n+1,n+1}} \;=\; -2\kappa\omega_{n+1,n+1} - \frac{n+2}{2}\kappa\varepsilon + \frac{1}{n+1}.$$

Knowing $A_n^\star$ exactly, we infer that the value of $\omega_{n+1,n+1}$ that maximizes the expectation $\mathcal{E}^{(2)}(A_n^\star, \omega_{n+1,n+1})$ is

$$\begin{aligned}\omega_{n+1,n+1} \;=\; \min\Big\{&\tfrac{1}{2(n+1)}X - \tfrac{1}{4}(n+2)\varepsilon, \\ &W_{(\text{ttl})} - \frac{n}{n+1}X - \frac{n}{2}\varepsilon\Big\}.\end{aligned}$$

The second term dominates this minimization whenever

$$W_{(\text{ttl})} \;\geq\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon;$$

therefore, if $W_{(\mathrm{ttl})}$ is large enough—as delimited by the preceding inequality—then

$$\begin{aligned}\mathcal{E}^{(2)}(A_n^\star, \omega_{n+1,n+1}) \;=\; & \\ & \frac{2n^2+2n+1}{4(n+1)^2}X - \frac{2n^2+3n+2}{4(n+1)}\varepsilon \\ & \qquad + \frac{(n+2)(2n^2+5n+6}{48}\kappa\varepsilon^2,\end{aligned}$$

When $W_{(\mathrm{ttl})}$ does not achieve this threshold, then

$$\begin{aligned}\mathcal{E}^{(2)}(A_n^\star, \omega_{n+1,n+1}) = & - W_{(\mathrm{ttl})}^2\kappa \\ & + \left(\frac{n-2}{2}\kappa\varepsilon + \frac{2n+1}{n+1}\right)W_{(\mathrm{ttl})} \\ & + \frac{(n^2+3n+14)n\kappa\varepsilon^2}{24} \\ & - \frac{n^2}{n+1}\varepsilon - \frac{n}{2(n+1)}X.\end{aligned}$$

For the found solution to be meaningful, the $(n+1)$th chunk must be nonempty, i.e., $\omega_{n+1,n+1} > 0$. This has two implications.

1) $X > \frac{(n+1)(n+2)}{2}\varepsilon$, which is true as long as $n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon} - 3\right)\right\rfloor$.
2) $W_{(\mathrm{ttl})} - (n/(n+1))X - \frac{1}{2}n\varepsilon > 0$, which implies $W_{(\mathrm{ttl})} > \frac{1}{2}n(n+1)\varepsilon$ because $X \geq W_{(\mathrm{ttl})}$. This inequality on $W_{(\mathrm{ttl})}$ is true as long as $n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\mathrm{ttl})}/\varepsilon} - 1\right)\right\rfloor$.

Because $X > \frac{1}{2}(n+1)(n+2)\varepsilon$, we have

$$\begin{aligned}& A_n + \omega_{n+1,n+1} + (n+1)\varepsilon \\ & \quad \leq \tfrac{n}{n+1}X - \tfrac{n}{2}\varepsilon + \tfrac{1}{2(n+1)\kappa} - \tfrac{1}{4}(n+2)\varepsilon + (n+1)\varepsilon \\ & \quad \leq X.\end{aligned}$$

For both Case 1 and Case 2, if either condition

$$\left[n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon} - 3\right)\right\rfloor\right]$$

or

$$\left[n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\mathrm{ttl})}/\varepsilon} - 1\right)\right\rfloor\right]$$

does not hold, then there is no optimal schedule with $(n+1)$ nonempty chunks. (We will come back later to the case where one of these conditions does not hold.) If both conditions hold, then Case 1 always has an optimal schedule, but Case 2 may not have one.

To complete the proof, we must verify that the optimal regimen always corresponds to Case 1 (as suggested by the theorem), never to Case 2 (whenever Case 2 defines a valid solution). We accomplish this by considering two cases, depending on the size $W_{(\mathrm{ttl})}$ of the workload. We show that the expected work completed under the regimen of Case 1 is never less than under the regimen of Case 2.

Case A: $W_{(\mathrm{ttl})} \geq \dfrac{n+1}{n+2}X - \dfrac{n+1}{2}\varepsilon$.

Under this hypothesis, and under Case 1, the workload that is actually deployed has size

$$W_{(\mathrm{dpl})} \;=\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,$$

so that, in expectation,

$$\begin{aligned}\mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1) = & \\ & W_{(\mathrm{dpl})} - \frac{n+1}{2n}W_{(\mathrm{dpl})}^2\kappa - \frac{n+1}{2}W_{(\mathrm{dpl})}\varepsilon\kappa \\ & + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa \\ & = \frac{n+1}{2(n+2)}X - \frac{n+1}{2}\varepsilon + \frac{(n+1)(n+2)(n+3)}{24}\varepsilon^2\kappa.\end{aligned}$$

units of work are completed. Moreover, because

$$\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon \;\leq\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon,$$

the most favorable value for $\mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1)$ under Case 2 lies within the range of values for the current case. Because the value of $\mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1)$ is constant whenever

$$W_{(\mathrm{ttl})} \;\geq\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,$$

we can reach the desired conclusion by just showing that this value is no smaller than: $\mathcal{E}^{(2)}\left(\dfrac{2n+1}{2n+2}X + \dfrac{n-2}{4}\varepsilon, n+1\right)$. Thus, we need only focus on the specific value

$$W_{(\mathrm{ttl-lim})} \;=\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon$$

for $W_{(\mathrm{ttl})}$. For this value, we have:

$$\begin{aligned}\mathcal{E}^{(2)}(W_{(\mathrm{ttl-lim})}, n+1) = & \\ & - W_{(\mathrm{ttl-lim})}^2\kappa + \left(\frac{n-2}{2}\kappa\varepsilon + \frac{2n+1}{n+1}\right)W_{(\mathrm{ttl-lim})} \\ & \quad + \frac{(n^2+3n+14)n\kappa\varepsilon^2}{24} - \frac{n^2}{n+1}\varepsilon - \frac{n}{2(n+1)}X \\ & = \frac{2n^2+2n+1}{4(n+1)^2}X - \frac{2n^2+3n+2}{4(n+1)}\varepsilon \\ & \quad + \frac{(n+2)(2n^2+5n+6)}{48}\varepsilon^2\kappa.\end{aligned}$$

By explicit calculation, we finally see that

$$\begin{aligned}& \mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1) - \mathcal{E}^{(2)}(W_{(\mathrm{ttl-lim})}, n+1) \\ & \quad = \frac{\left(4 + (n^4+6n^3+13n^2+12n+4)\kappa^2\varepsilon^2\right)n}{16(n+1)^2(n+2)\kappa} \\ & \qquad - \frac{(4n^2+12n+8)\kappa\varepsilon n}{16(n+1)^2(n+2)\kappa} \\ & \quad = \frac{\left(4 + (n+1)^2(n+2)^2\kappa^2\varepsilon^2 - 4(n+1)(n+2)\kappa\varepsilon\right)n}{16(n+1)^2(n+2)\kappa} \\ & \quad = \frac{\left((n+1)(n+2)\kappa\varepsilon - 2\right)^2 n}{16(n+1)^2(n+2)\kappa} \\ & \quad \geq 0.\end{aligned}$$

Case B: $W_{(\mathrm{ttl})} \leq \dfrac{n+1}{n+2}X - \dfrac{n+1}{2}\varepsilon$.

In this case, the regimen of Case 1 deploys all $W_{(\mathrm{ttl})}$ units of work, thereby completing, in expectation,

$$\mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1) = W_{(\mathrm{ttl})} - \frac{n+1}{2n}W_{(\mathrm{ttl})}^2\kappa - \frac{n+1}{2}W_{(\mathrm{ttl})}\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa.$$

units of work. Moreover,

$$\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon \;\leq\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon,$$

so that the regimen of Case 2 also deploys all $W_{(\mathrm{ttl})}$ units of work, thereby completing, in expectation,

$$\mathcal{E}^{(2)}(W_{(\mathrm{ttl})}, n+1) = - W_{(\mathrm{ttl})}^2\kappa + \left(\frac{n-2}{2}\kappa\varepsilon + \frac{2n+1}{n+1}\right)W_{(\mathrm{ttl})} + \frac{(n^2+3n+14)n\kappa\varepsilon^2}{24} - \frac{n^2}{n+1}\varepsilon - \frac{n}{2(n+1)}X.$$

units of work.

Explicit calculation now shows that

$$\mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1) - \mathcal{E}^{(2)}(W_{(\mathrm{ttl})}, n+1) = \frac{n}{2(n+1)}W_{(\mathrm{ttl})}^2\kappa - \frac{n}{n+1}(1+(n+1)\varepsilon\kappa)W_{(\mathrm{ttl})} + \frac{n}{2(n+1)}(1+2n\kappa\varepsilon-(n+1)\kappa^2\varepsilon^2)X.$$

Viewed as a function of $W_{(\mathrm{ttl})}$, this difference is, thus, unimodal, decreasing up to its global minimum, which occurs at $W_{(\mathrm{ttl})} = X+(n+1)\varepsilon$, and increasing thereafter. The largest value of $W_{(\mathrm{ttl})}$ allowed by the current case is

$$W_{(\mathrm{ttl}-\max)} \;=\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,$$

so this is also the value on which the difference $\mathcal{E}^{(1)}(W_{(\mathrm{ttl})}, n+1) - \mathcal{E}^{(2)}(W_{(\mathrm{ttl})}, n+1)$ reaches its minimum within its domain of validity. Thus, we need only focus on the behavior of the difference at the value $W_{(\mathrm{ttl})} = W_{(\mathrm{ttl}-\max)}$. At this value,

$$\mathcal{E}^{(1)}(W_{(\mathrm{ttl}-\max)}, n+1) - \mathcal{E}^{(2)}(W_{(\mathrm{ttl}-\max)}, n+1) = \frac{n(5n+1)\varepsilon^2\kappa}{8} + \frac{(n-1)n\varepsilon}{2(n+1)(n+2)} + \frac{n}{2(n+1)(n+2)^2\kappa}.$$

This quantity is obviously positive, which means that $\mathcal{E}^{(1)}(W_{(\mathrm{ttl}-\max)}, n+1) > \mathcal{E}^{(2)}(W_{(\mathrm{ttl}-\max)}, n+1)$.

We thus see that, for workloads of any size $W_{(\mathrm{ttl})}$, one completes at least as much expected work via the schedule of Case 1 as via the schedule of Case 2.

In summation, if

$$n \;\leq \min\left\{\begin{array}{l}\left\lfloor\frac{1}{2}\left(\sqrt{1+8X/\varepsilon}-3\right)\right\rfloor \\ \left\lfloor\frac{1}{2}\left(\sqrt{1+8W_{(\mathrm{ttl})}/\varepsilon}-1\right)\right\rfloor\end{array}\right. \tag{12}$$

then Case 1 specifies the optimal schedule that uses no more than $n+1$ chunks. Of course, this inequality translates to the conditions of the theorem (where it is written for n chunks instead of $n+1$).

Note that if n exceeds either quantity in the minimization of (12), then one never improves the expected amount of work completed by deploying the workload in more than n chunks. This is another consequence of our remark about A_n at the beginning of this proof. If there exists a value of m for which there exists a schedule $\mathcal{S}$ that uses $\geq n+1$ nonempty chunks, then replacing the first $n+1$ chunks in this solution with the optimal solution for n chunks, using a workload equal to the first $n+1$ chunks of $\mathcal{S}$, yields a schedule that, in expectation, completes strictly more work than $\mathcal{S}$. □

About arbitrary risk. Even with one single computer, we could obtain only asymptotic optimality results under arbitrary risk with the free-initiation model (see Section 3.1.2). Using Theorem 1, we can still derive an asymptotic optimal schedule for the charged-initiation model.

4 Scheduling Two Remote Computers with Free Initiation

When scheduling a single remote computer, one needs not cope with many of the hardest aspects of scheduling many remote computers, notably those involving replicating and ordering work. In contrast, when scheduling *two* remote computers, one does have to cope with such issues. Accordingly, our work in this section allows us to derive principles that guide us as we address the general case of p remote computers in the next section. We focus in this section only on the free-initiation model, in order to simplify the problem of work replication. By Theorem 1, the optimal schedules we derive are asymptotically optimal under the charged-initiation model.

We begin this section with a result that narrows the search for optimal schedules by identifying three characteristics that we may insist on in any candidate optimal schedule for two remote computers (Section 4.1). We then observe these characteristics "in action" as we derive explicit schedules that are asymptotically optimal under arbitrary risk functions (Section 4.2). We finally refine the latter schedules for the linear risk model, developing schedules that are always at least asymptotically optimal and are exactly optimal under certain circumstances (Section 4.3). A major message from this section is that it is much harder to share work optimally with multiple computers than with a single computer. We therefore devote the next section, which deals with scheduling arbitrary numbers of remote computers, to a quest for well-structured heuristics whose quality is established experimentally.

4.1 Guidelines for Crafting Optimal Schedules

Say that we are scheduling two remote computers, P_1 and P_2. Say further that for $j = 1, 2$, we deploy n_j chunks of work, call them $\mathcal{W}_{j,1}, \ldots, \mathcal{W}_{j,n_j}$, on computer P_j, and

that P_j must schedule its chunks in the indicated order. We do not assume any *a priori* relation between how P_1 and P_2 break their allocated work into chunks; in fact, work that is allocated to both P_1 and P_2 (what we later call "replicated" work) may be chunked differently on the two computers. Absent more information about the risk function that governs the current worksharing episode, we are not yet ready to prescribe what is the best way to schedule the work on P_1 and P_2; however, we are able to provide valuable guidelines for this scheduling.

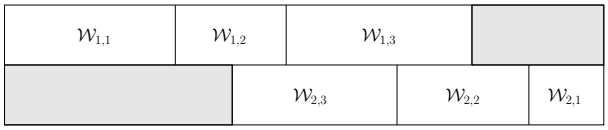


Fig. 1. The shape of an optimal schedule for two computers; see Theorem 5 with $n_1 = n_2 = 3$. The top row displays P_1's chunks, the bottom row P_2's. Vertically aligned parts of chunks correspond to replicated work; shaded areas depict unallocated work (e.g., none of the work in $\mathcal{W}_{2,1}$ is allocated to P_1).

Theorem 5: (Guidelines: nondecreasing risk) We want to schedule a pair of computers, P_1 and P_2, whose common probability of being interrupted never decreases as a computer processes more work. Given any schedule Σ for the computers, there exists a schedule Σ' that, in expectation, completes as much work as Σ does and that enjoys the following characteristics (cf. Fig. 1).

1) *Maximal work deployment.* Σ' deploys as much of the overall workload as possible.
   In particular, the workloads that Σ' deploys to P_1 and P_2 can overlap only if their union is the entire overall workload.
2) *Local-work prioritization.* Σ' has P_1 (resp., P_2) process all of the allocated work that is *not* replicated on P_2 (resp., on P_1) before it processes any replicated work.
3) *"Mirroring" of replicated work.* Σ' has P_1 and P_2 process their copies of replicated work "in opposite orders."
   In detail, say that, for $j = 1, 2$, computer P_j chops its allocated work into chunks $\mathcal{W}_{j,1}, \ldots, \mathcal{W}_{j,n_j}$. Say that there exist chunk-indices a_1 and $b_1 > a_1$ for P_1, and a_2 and $b_2 > a_2$ for P_2 such that: chunks $\mathcal{W}_{1,a_1}$ and $\mathcal{W}_{2,a_2}$ both contain copies of a replicated "piece of work" A, and chunks $\mathcal{W}_{1,b_1}$ and $\mathcal{W}_{2,b_2}$ both contain copies of a replicated "piece of work" B. Then if Σ' has P_1 execute A before B, then Σ' has P_2 execute B before A.

Note that, because our schedules are chunk-oriented, the phrase "having P_1 execute A before B" really means having P_1 execute chunk $\mathcal{W}_{1,a_1}$ before chunk $\mathcal{W}_{1,b_1}$; similarly, "having P_2 execute B before A" really means having P_2 execute chunk $\mathcal{W}_{2,b_2}$ before chunk $\mathcal{W}_{2,a_2}$.

Proof: The strategy. We devise a cut-and-paste argument for each of the theorem's three characteristics in turn. Each time, we begin with an arbitrary schedule Σ that does not have that characteristic, and we show how to alter Σ to a schedule Σ' that does have the characteristic and that, in expectation, completes as much work as does Σ. In order to achieve the required alterations, we must refine our description of workloads. Specifically, we now describe the overall workload via a partition $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ of *pieces of work*, that has the following property. For $j = 1, 2$, each piece $\mathcal{X}_i \in \mathcal{X}$ is either included within a chunk of P_j, or it is disjoint from each chunk of P_j. We define, for $j = 1, 2$ and $i = 1, \ldots, m$, the indicator

$$\delta_j(i) = \begin{cases} 0 & \text{when } \mathcal{X}_i \in \mathcal{X} \text{ does not intersect any chunk of } P_j \\ 1 & \text{when } \mathcal{X}_i \text{ is contained within a chunk of } P_j, \text{ say, chunk } \sigma_j(i) \\ & \text{— so that } \mathcal{X}_i \subset \mathcal{W}_{j,\sigma_j(i)} \end{cases}$$

We now specify the expectation, $\mathcal{E}$, of $\mathsf{W}_{(\mathrm{cmp})}$ under this new specification of the deployment of chunks to P_1 and P_2. The probability that piece $\mathcal{X}_i \in \mathcal{X}$ is computed successfully is:

- 0 if $\mathcal{X}_i$ is not allocated to either P_1 or P_2;
- $1 - Pr\left(\sum_{j=1}^{\sigma_k(i)} \omega_{k,j}\right)$ if $\mathcal{X}_k$ is allocated only to P_k i.e., if $\delta_k(i)(1 - \delta_{k'}(i)) = 1$, where $\{k, k'\} = \{1, 2\}$;
- $1 - Pr\left(\sum_{j=1}^{\sigma_1(i)} \omega_{1,j}\right) Pr\left(\sum_{j=1}^{\sigma_2(i)} \omega_{2,j}\right)$ if $\mathcal{X}_i$ is allocated to both P_1 and P_2; i.e., if $\delta_1(i)\delta_2(i) = 1$.

We thus have

$$\mathcal{E} = \sum_{i=1}^{m} |\mathcal{X}_i| \cdot \Xi_i \tag{13}$$

where

$$\begin{aligned} \Xi_i \;=\; & \delta_1(i)\delta_2(i) \left(1 - Pr\left(\sum_{j=1}^{\sigma_1(i)} \omega_{1,j}\right) Pr\left(\sum_{j=1}^{\sigma_2(i)} \omega_{2,j}\right)\right) \\ & + \delta_1(i)(1 - \delta_2(i)) \left(1 - Pr\left(\sum_{j=1}^{\sigma_1(i)} \omega_{1,j}\right)\right) \\ & + (1 - \delta_1(i))\delta_2(i) \left(1 - Pr\left(\sum_{j=1}^{\sigma_2(i)} \omega_{2,j}\right)\right). \end{aligned}$$

The alterations. We now look at each of our three characteristics in turn, performing the following process for each. We begin with a schedule $\Sigma^{(0)}$ that, in expectation, completes $E^{(0)}$ units of work. Say, for induction, that we now have a schedule $\Sigma^{(r)}$ that completes $E^{(r)}$ units of work. We describe how to alter $\Sigma^{(r)}$ to obtain a schedule $\Sigma^{(r+1)}$ that, in a sense, comes closer to enjoying the current characteristic and that, in expectation, completes $E^{(r+1)} \geq E^{(r)}$ units of work. We prove that a finite sequence of such alterations converts $\Sigma^{(0)}$ to a schedule Σ that enjoys the characteristic.

Maximal work deployment. Say that schedule Σ deploys some portion of the overall workload to both P_1 and P_2, while it leaves some other piece unallocated to either:

- the doubly allocated portion is a piece $\mathcal{X}_i \in \mathcal{X}$;

- the unallocated work is a piece $\mathcal{X}_j \in \mathcal{X}$.

To alter schedule Σ, we wish somehow to swap some doubly allocated work for an equal amount of unallocated work. This is easy when $|\mathcal{X}_i| = |\mathcal{X}_j|$. Otherwise, we achieve this goal as follows.

1) If $|\mathcal{X}_j| < |\mathcal{X}_i|$, then we invoke divisibility to subdivide $\mathcal{X}_i$ into one piece, $\mathcal{A}$, of size $|\mathcal{X}_j|$ and another of size $|\mathcal{X}_i| - |\mathcal{A}|$. We swap $\mathcal{B} = \mathcal{X}_j$ for $\mathcal{A}$ in the chunk of P_1 that contains $\mathcal{X}_i$.
2) If $|\mathcal{X}_j| > |\mathcal{X}_i|$, then we invoke divisibility to subdivide $\mathcal{X}_j$ into one piece, $\mathcal{B}$, of size $|\mathcal{X}_i|$ and another of size $|\mathcal{X}_j| - |\mathcal{B}|$. We swap $\mathcal{B}$ for $\mathcal{A} = \mathcal{X}_i$ in the relevant chunk of P_1.

In case 1, $\Sigma^{(r+1)}$ has no more unallocated work and the maximal deployment rule is in force. In case 2, $\Sigma^{(r+1)}$ has one fewer piece of doubly allocated work. It follows that a finite sequence of alterations convert $\Sigma^{(0)}$ into a schedule that practices maximal work deployment. *We henceforth assume that Σ practices maximal work deployment.*

Local-work prioritization. Assume that under optimal schedule Σ, there exist $\mathcal{X}_i, \mathcal{X}_j \in \mathcal{X}$ such that:

1) Σ allocates $\mathcal{X}_i$ to P_1 but not to P_2; i.e., $\delta_1(i)(1 - \delta_2(i)) = 1$;
2) Σ allocates $\mathcal{X}_j$ to both P_1 and P_2; i.e., $\delta_1(i)\delta_2(i) = 1$;
3) Σ attempts to execute $\mathcal{X}_j$ before $\mathcal{X}_i$ on P_1: symbolically, $\sigma_1(i) \geq \sigma_1(j)$.

We alter Σ to obtain a new schedule that comes closer to prioritizing local work. And, we do so in a way that (a) at least matches Σ's expected work production and (b) guarantees that a finite sequence of alterations produce a schedule that practices local work prioritization. We proceed as follows.

We codify the set of violations of local work prioritization via the set $\mathcal{V}$ that identifies every triplet of chunks that violate local work prioritization.

$$\mathcal{V} = \left\{ (k,k',l) \;\middle|\; \begin{array}{l} \mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'} \neq \emptyset \text{ (replicated work)} \\ \mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'} \neq \emptyset \text{ (local work)} \\ k < l \text{ (replicated starts before local)} \\ k \in [1,n_1], k' \in [1,n_2], l \in [1,n_1] \end{array} \right\}$$

To choose the alteration to apply to Σ at this step, we take any triplet $(k,k',l) \in \mathcal{V}$ whose first component, k, is minimal among all the first components of elements of $\mathcal{V}$, and whose third component, l, is maximal among all the third components of elements of $\mathcal{V}$ (one verifies easily that such an element always exists). From the perspective of P_1, we thus focus on the earliest-scheduled chunk containing a replicated piece of work that is scheduled before the latest-scheduled chunk that contains an unreplicated piece of work.

(1) Say first that $|\mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}| \leq |\mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}|$. In this case, we alter Σ by swapping the piece of work $\mathcal{A} = \mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}$ from $\mathcal{W}_{1,l}$ with an arbitrarily chosen like-sized subset $\mathcal{B}$ of $\mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}$ in $\mathcal{W}_{1,k}$. After the swap, $\mathcal{V}$ no longer contains any element of the form (α,β,l), because chunk $\mathcal{W}_{1,l}$ now contains only replicated work. Furthermore, by choice of k, chunks $\mathcal{W}_{1,1}$ through $\mathcal{W}_{1,k-1}$ contain only work for P_1 that is not replicated on P_2. Thus, the swap reduces the number of violations.

(2) Say alternatively that $|\mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}| > |\mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}|$. In this case, we alter Σ by swapping the piece of work $\mathcal{B} = \mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}$ from $\mathcal{W}_{1,k}$ with an arbitrarily chosen like-sized subset $\mathcal{A}$ of $\mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}$ in $\mathcal{W}_{1,l}$. After the swap, $\mathcal{V}$ no longer contains any element of the form (k,k',α). Furthermore, by definition of k, chunks $\mathcal{W}_{1,1}$ through $\mathcal{W}_{1,k-1}$ contain only work for P_1 that is not replicated on P_2. Thus, this swap also reduces the number of violations.

Clearly, at most $|\mathcal{V}|$ alterations are needed to convert Σ to a schedule that practices local-work prioritization *on computer* P_1. Because each alteration affects only the scheduling on P_1, we can now apply an analogous sequence of alterations that focus on violations by computer P_2. After this second round of alterations, we have finally converted Σ to a schedule that practices local work prioritization on *both* computers. *We henceforth assume that Σ practices local-work prioritization.*

"Mirroring" of replicated work. Say that, under schedule Σ, there are two partition elements, $\mathcal{X}_i$ and $\mathcal{X}_j$, such that:

1) Σ allocates both $\mathcal{X}_i$ and $\mathcal{X}_j$ to both P_1 and P_2; i.e., $\delta_1(i)\delta_2(i) = \delta_1(j)\delta_2(j) = 1$;
2) Σ attempts to execute $\mathcal{X}_i$ after $\mathcal{X}_j$ on both P_1 and P_2: symbolically, $[\sigma_1(i) \geq \sigma_1(j)]$ and $[\sigma_2(i) \geq \sigma_2(j)]$.

We craft a sequence of alterations to Σ that produce a schedule that practices the mirroring of replicated work. Essentially, at each step, we identify a pair of pieces of work, $\mathcal{A}$ and $\mathcal{B}$, that violate mirroring in the way just described. We then swap $\mathcal{B}$ for $\mathcal{A}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$ and swap $\mathcal{A}$ for $\mathcal{B}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{B})}$, while leaving the schedule of P_2 unchanged.

How do we select the pieces to focus on at this step? Our job is somewhat simplified by our ability to focus entirely on replicated pieces of work—because of our assumption that Σ practices both maximal work deployment and local-work prioritization. We employ the following inductive process to choose the pieces from among pieces of replicated work that violate mirroring. Say, for induction, that we have so far altered Σ k times—so that the k pieces of replicated work that P_1 is scheduled to (attempt to) execute *first* are the k pieces of replicated work that P_2 is scheduled to (attempt to) execute *last*, and that these pieces are executed in reverse orders on P_1 and P_2. We now select the $(k+1)$th piece of replicated work that P_1 is scheduled to (attempt to) execute, call it $\mathcal{X}_i$, and the kth from last piece of replicated work that P_2 is scheduled to (attempt to) execute, call it $\mathcal{X}_j$.

(1) If $\mathcal{X}_i = \mathcal{X}_j$, then there is no violation to undo.

(2) If $\mathcal{X}_i \neq \mathcal{X}_j$, then we select the pieces, $\mathcal{A}$ and $\mathcal{B}$, to swap in the $(k+1)$th alteration of Σ, in the following manner. (a) If $|\mathcal{X}_j| \geq |\mathcal{X}_i|$, then we have $\mathcal{X}_i$ play the role of $\mathcal{A}$, and we select as $\mathcal{B}$ any size-$|\mathcal{A}|$ subset of $\mathcal{X}_j$. After the swap, $\mathcal{B}$ is scheduled as the $(k+1)$th piece of replicated work for P_1 to (attempt to) execute and

(in deference to mirroring) as the kth from last piece of replicated work for P_2 to (attempt to) execute. None of the first k pieces of replicated work for P_1 nor any of the last k pieces of replicated work for P_2 were affected by the swap. To conclude that the inductive process eventually terminates (successfully!) we first consider the number of chunks of P_1 (respectively P_2) that include only work from the first (resp. last) k pieces of replicated work. As the alterations progress, the numbers of such pieces never decrease. If an alteration does not increase either of these numbers, then we focus on the set of early chunks of P_2 that contain work that is replicated in the chunk of P_1 that we are working with:

$$\mathcal{V}_2 = \left\{ l < \sigma_2(\mathcal{X}_j) \mid \mathcal{W}_{1,\sigma_1(\mathcal{X}_i)} \cap \mathcal{W}_{2,l} \neq \emptyset \right\},$$

and the symmetrical set of chunks for P_1:

$$\mathcal{V}_1 = \left\{ l > \sigma_1(\mathcal{X}_i) \mid \mathcal{W}_{1,l} \cap \mathcal{W}_{2,\sigma_2(\mathcal{X}_j)} \neq \emptyset \right\}.$$

If we assume—as we may with no loss of generality—that we are working with a partition made of *maximal* elements, then the alteration decreases the set $\mathcal{V}_2$ by one element (namely, $\mathcal{W}_{2,\sigma_2(\mathcal{X}_i)}$) and does not modify the set $\mathcal{V}_1$. (b) The case when $|\mathcal{X}_{j'}| < |\mathcal{X}_{i'}|$ is symmetric with case (a), hence is left to the reader. We note only that, in that case, the alteration does not modify the set $\mathcal{V}_2$, and it decreases the set $\mathcal{V}_1$ by one element (namely, $\mathcal{W}_{1,\sigma_1(\mathcal{X}_j)}$)

Validating the alterations. To complete the proof, we need only verify that each of the schedule alterations we have described cannot produce a schedule that completes less work than schedule Σ does. An important feature of our alterations that greatly simplifies these verifications is that, in each case, the relevant alteration affects only the terms in expression (13) that mention the pieces involved in the alteration.

Maximal work deployment. Recall that our alteration of schedule Σ in this case substituted piece $\mathcal{B}$ for piece $\mathcal{A}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$. Now, before this substitution, the total contribution to the expectation $\mathcal{E}$ of these pieces was:

$$|\mathcal{A}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \cdot 0.$$

After the substitution, this contribution becomes:

$$|\mathcal{A}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right).$$

Because $|\mathcal{A}| = |\mathcal{B}|$, the latter contribution is never less than the former, differing from it by the quantity

$$|\mathcal{A}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right) \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right),$$

whose nonnegativity implies that the altered schedule completes at least as much work, in expectation, as does schedule Σ.

Local work prioritization. Recall that our alteration of schedule Σ in this case substituted a piece of local work $\mathcal{A}$ from $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$ with a piece of replicated work $\mathcal{B}$ of $\mathcal{W}_{1,\sigma_1(\mathcal{B})}$. Now, before this substitution, the total contribution to the expectation $\mathcal{E}$ of these pieces was:

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

After the substitution, the contribution becomes

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

Because $|\mathcal{A}| = |\mathcal{B}|$, we see that the substitution increases the overall expectation by the quantity

$$|\mathcal{A}| \times \left(Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right)\right) \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

This quantity is nonnegative because

- the probability $Pr(w)$ is nondecreasing in w;
- $\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k} \geq \sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}$, because $\sigma_1(\mathcal{A}) \geq \sigma_1(\mathcal{B})$.

The altered schedule completes, in expectation, at least as much work as Σ.

Shared work "mirroring." Recall that our alteration of schedule Σ in this case swapped $\mathcal{B}$ for $\mathcal{A}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$ and swapped $\mathcal{A}$ for $\mathcal{B}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{B})}$, while leaving the schedule of P_2 unchanged. Now, before this substitution, the total contribution to the expectation $\mathcal{E}$ of these pieces was:

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

After the substitution, their contribution becomes:

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

Because $|\mathcal{A}| = |\mathcal{B}|$, the substitutions have increased the overall expectation by the quantity:

$$|\mathcal{A}| \times \left(Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right) \times \left(Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right) - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right).$$

This quantity is nonnegative because $\mathcal{A}$ and $\mathcal{B}$ are processed in the same order on P_1 and on P_2. The altered schedule thus completes, in expectation, at least as much work as Σ.

□

4.2 Two Remote Computers under Arbitrary Risk

Our results in [1] suggest that finding exactly optimal schedules for two remote computers is surprisingly difficult, even in the simplest case wherein each computer processes its allocated work as a single chunk or when the entire workload is deployed on each computer. The simulations in those sources also suggest that simple regular solutions often complete, in expectation, almost as much work as do complex exactly optimal ones. These results lead us to abandon our quest for exactly optimal schedules, in favor of asymptotically optimal schedules with simple structure. Indeed, if one is willing to settle for *asymptotically optimal* expected work-production, then reasoning analogous to that used in analyzing one remote computer shows how to adapt the asymptotically optimal schedule $\Sigma(n)$ of Theorem 3—in the light of the guidelines of Theorem 5—to produce asymptotically optimal expected work-production with two remote computers.

Theorem 6: (Asymptotically optimal schedule: arbitrary risk) One wishes to deploy $W_{(\mathrm{ttl})}$ units of work to two remote computers, in $\leq n$ chunks, for some $n > 0$.
The goal. To maximize the expectation of $\mathsf{W}_{(\mathrm{cmp})}$.
An asymptotically optimal schedule. One allocates the same set of equal-size chunks to both computers, in the following "mirrored" manner:

$$(\forall i \in [1,n])\left[\mathcal{W}_{1,i}, \mathcal{W}_{2,n-i+1} \leftarrow \left\langle \frac{i-1}{n} W_{(\mathrm{ttl})}, \frac{i}{n} W_{(\mathrm{ttl})} \right\rangle\right].$$

Proof: In this proof, we denote by $\mathcal{S}(n)$ the regimen using n chunks we want to establish the asymptotic optimality of. We denote by $\Sigma(n)$ an optimal regimen using (at most) n chunks.

Thanks to Theorem 5, we know the general shape of schedule $\Sigma(n)$. Without loss of generality, we can indeed assume that $\Sigma(n)$ has the shape described on Figure 2(a). We thus begin—see Fig. 2(a)—with an optimal schedule $\Sigma(n)$ that processes work in n chunks and that satisfies the three properties of Theorem 5. First—see Fig. 2(b)—we transform $\Sigma(n)$ to schedule $\Sigma_1(n)$, by adding a possibly empty $(n+1)$th chunk to the workload of each computer so that each computer processes the entire workload. Clearly, this transformation cannot decrease expected work production. Next—see Fig. 2(c)—we transform $\Sigma_1(n)$ to schedule $\Sigma_2(n)$ by subdividing chunks so that both computers' chunk boundaries coincide. We accomplish this as follows. Focus on the $(n+1)$-element increasing sequences, $\vec{\mathcal{B}}_1$ and $\vec{\mathcal{B}}_2$, of "places" in the workloads of computers P_1 and P_2, respectively, at which the chunks end:

$$\begin{aligned}\vec{\mathcal{B}}_1 &= \omega_{1,1},\ (\omega_{1,1}+\omega_{1,2}), \ldots,\ \sum_{j=1}^{n+1}\omega_{1,j}\\ \vec{\mathcal{B}}_2 &= W_{(\mathrm{ttl})} - \sum_{j=1}^{n+1}\omega_{2,j},\ W_{(\mathrm{ttl})} - \sum_{j=1}^{n}\omega_{2,j},\\ &\quad \ldots, W_{(\mathrm{ttl})} - (\omega_{2,1}+\omega_{2,2}),\ W_{(\mathrm{ttl})} - \omega_{2,1}\end{aligned}$$

Merge the elements of $\vec{\mathcal{B}}_1$ and $\vec{\mathcal{B}}_2$ into an increasing sequence, $b_0 < b_1 < \cdots < b_l = W_{(\mathrm{ttl})}$; clearly, $l \leq 2n+1$, because $\vec{\mathcal{B}}_1$ and $\vec{\mathcal{B}}_2$ could contain many elements in common and *must* both end with $W_{(\mathrm{ttl})}$. Then specify the new, coterminal, chunks by partitioning $W_{(\mathrm{ttl})}$ into l chunks as follows:

$$\mathcal{W}'_{1,i} = \mathcal{W}'_{2,l-i+1},\ \text{where each } \omega'_{1,i} = b_i - b_{i-1}.$$

Finally, we remark that subdividing chunks does not decrease the overall expected work production. Therefore:

$$E(W_{(\mathrm{ttl})}, \Sigma(n)) \leq E(W_{(\mathrm{ttl})}, \Sigma_2(n)).$$

Let us now consider any strictly positive integer m. We are going to compare the performance of the scheduling regimens $\Sigma_2(n)$ and $\mathcal{S}(m)$.

For that purpose, we introduce three more notations. First, we denote by α the size of a chunk of $\mathcal{S}(m)$: $\alpha = \frac{W_{(\mathrm{ttl})}}{m}$. Then, for any $i \in [1..m-1]$, let $s(i)$ be the index of the first chunk of $\mathcal{S}(m)$ which starts no sooner than the end of the i-th chunk of $\Sigma_2(n)$ (on computer P_1). Formally:

$$s(i) = 1 + \left\lceil \frac{\sum_{k=1}^{i} \omega'_{1,i}}{\alpha} \right\rceil.$$

Symmetrically, for any $i \in [1..m]$, let $p(i)$ be the index of the last chunk of $\mathcal{S}(m)$ which ends no later than the beginning of the i-th chunk of $\Sigma_2(n)$ (on computer P_1). Formally:

$$p(i) = \left\lfloor \frac{\sum_{k=1}^{i-1} \omega'_{1,i}}{\alpha} \right\rfloor.$$

The overall expectation of $\mathsf{W}_{(\mathrm{cmp})}$ for $\Sigma_2(n)$ is:

$$E(W_{(\mathrm{ttl})}, \Sigma_2(n)) = \sum_{i=1}^{l} \omega'_{1,i}\left(1 - Pr\left(\sum_{j=1}^{i}\omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1}\omega'_{1,j}\right)\right). \quad (14)$$

Let us now consider any chunk $\mathcal{W}'_{1,i}$ of $\Sigma_2(n)$ (that is, any $i \in [1..m]$). Its contribution to the overall expectation

is:

$$e_i = \omega'_{1,i}\left(1 - Pr\left(\sum_{j=1}^{i}\omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1}\omega'_{1,j}\right)\right). \quad (15)$$

If $\omega'_{1,i} < 2\alpha$, obviously $e_i < 2\alpha$. Otherwise, $\omega'_{1,i} \geq 2\alpha$ and there exists at least one chunk of $\mathcal{S}(m)$ which is included in the chunk $\mathcal{W}_{1,i}$. Then, $p(i+1) \geq s(i-1)$ (we extend s by letting $s(0) = 0$). We can then establish:

$$\begin{aligned}\omega'_{1,i} &= \sum_{j=1}^{i}\omega'_{1,j} - \sum_{j=1}^{i-1}\omega'_{1,j} \\ &= \left(\sum_{j=1}^{i}\omega'_{1,j} - p(i+1)\alpha\right) \\ &\quad + (p(i+1)\alpha - (s(i-1)-1)\alpha) \\ &\quad + \left((s(i-1)-1)\alpha - \sum_{j=1}^{i-1}\omega'_{1,j}\right) \\ &< \alpha + (p(i+1) - s(i-1) + 1)\,\alpha + \alpha.\end{aligned}$$

Using this result and Equation (15) we can bound the value of e_i:

$$\begin{aligned}e_i &< (2\alpha + (p(i+1) - s(i-1) + 1)\,\alpha) \\ &\quad \times \left(1 - Pr\left(\sum_{j=1}^{i}\omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1}\omega'_{1,j}\right)\right) \\ &\leq 2\alpha \\ &\quad + \sum_{j=s(i-1)}^{p(i+1)} \alpha\left(1 - Pr\left(\sum_{j=1}^{i}\omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1}\omega'_{1,j}\right)\right) \\ &\leq 2\alpha \\ &\quad + \sum_{j=s(i-1)}^{p(i+1)} \alpha\left(1 - Pr\,(j\alpha)\,Pr\left(W_{(\mathrm{ttl})} - (j-1)\alpha\right)\right).\end{aligned}$$

The last inequation holds because Pr is a non decreasing function, because $p(i+1)\alpha$ is no greater than $\sum_{j=1}^{i}\omega'_{1,j}$, and because $(s(i-1)-1)\alpha$ is no smaller than $\sum_{j=1}^{i-1}\omega'_{1,j}$.

We can now rewrite Equation (14):

$$\begin{aligned}&E(W_{(\mathrm{ttl})}, \Sigma_2(n)) \\ &= \sum_{\substack{1\leq i\leq l \\ \omega'_{1,i}<2\alpha}} e_i + \sum_{\substack{1\leq i\leq l \\ \omega'_{1,i}\geq 2\alpha}} e_i \\ &< \sum_{\substack{1\leq i\leq l \\ \omega'_{1,i}<2\alpha}} 2\alpha \\ &\quad + \sum_{\substack{1\leq i\leq l \\ \omega'_{1,i}\geq 2\alpha}} \left(2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha\left(1 - Pr(j\alpha)\,Pr\left(W_{(\mathrm{ttl})} - (j-1)\alpha\right)\right)\right) \\ &\leq 2l\alpha + \sum_{j=1}^{l} \alpha\left(1 - Pr\,(j\alpha)\,Pr\left(W_{(\mathrm{ttl})} - (j-1)\alpha\right)\right) \\ &\leq \frac{4n+2}{m} W_{(\mathrm{ttl})} + E(W_{(\mathrm{ttl})}, \mathcal{S}(m)).\end{aligned}$$

Therefore, for any positive integers n and m:

$$\begin{aligned}&E(W_{(\mathrm{ttl})}, \Sigma(n)) - \frac{4n+2}{m} W_{(\mathrm{ttl})} \\ &\quad \leq E(W_{(\mathrm{ttl})}, \Sigma_2(n)) - \frac{4n+2}{m} W_{(\mathrm{ttl})} \\ &\quad < E(W_{(\mathrm{ttl})}, \mathcal{S}(m)) \\ &\quad \leq E(W_{(\mathrm{ttl})}, \Sigma(m)). \quad (16)\end{aligned}$$

$E(W_{(\mathrm{ttl})}, \Sigma(n))$ is obviously a nondecreasing, upper-bounded (by $W_{(\mathrm{ttl})}$), sequence and it thus converges. By replacing n by $\lfloor\sqrt{m}\rfloor$ in Equation (16), one easily sees that the sequence $E(W_{(\mathrm{ttl})}, \mathcal{S}(m))$ is converging with the same limit. □

4.3 Two Remote Computers under Linear Risk

We finally specialize from arbitrary risk functions to linear risk, in order to get stronger performance guarantees under certain conditions.

4.3.1 Exact optimality for single-chunk deployment

Theorem 7: (Optimal schedule: linear risk, single-chunk deployment) One wishes to deploy $W_{(\mathrm{ttl})}$ units of work on two computers, deploying a single chunk on each computer. The following schedule Σ is optimal among *symmetric* schedules.

- If $W_{(\mathrm{ttl})} \leq \frac{1}{2}X$, then Σ deploys the entire workload on both computers; symbolically, $\mathcal{W}_{1,1} = \mathcal{W}_{2,1} = \langle 0, W_{(\mathrm{ttl})}\rangle$.
- If $\frac{1}{2}X < W_{(\mathrm{ttl})} \leq X$, then Σ deploys the first half of the workload on one computer and the second half on the other; symbolically, $\mathcal{W}_{1,1} = \langle 0, \frac{1}{2}W_{(\mathrm{ttl})}\rangle$, and $\mathcal{W}_{2,1} = \langle \frac{1}{2}W_{(\mathrm{ttl})}, W_{(\mathrm{ttl})}\rangle$.
- If $X < W_{(\mathrm{ttl})}$, then Σ deploys only X units of the workload, allocating the first half to one computer and the second half to the other; symbolically, $\mathcal{W}_{1,1} = \langle 0, \frac{1}{2}X\rangle$, and $\mathcal{W}_{2,1} = \langle \frac{1}{2}X, X\rangle$.

Proof sketch: We derive the optimal schedule in the light of the following principle (which we encountered before). When we deploy work to a remote computer as a single chunk, we should never make that chunk as large as X, for that size would risk certain interruption, hence, in expectation, complete no work. Our analysis distinguishes schedules that deploy disjoint workloads to the two computers from those that do not.

If schedule Σ deploys *disjoint* workloads to the two remote computers, then the independence of the workloads allows us to invoke Theorem 2 to discover their optimal sizes. The optimal strategy is to deploy $W_{(\mathrm{dpl})} = \min\{W_{(\mathrm{ttl})}, X\}$ units of work in total. If Σ deploys this work in chunks of respective sizes $\omega_{1,1}$ and $\omega_{2,1} = W_{(\mathrm{dpl})} - \omega_{1,1}$, then, by (1), the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ is

$$\mathcal{E} \;=\; -2\omega_{1,1}^2\kappa + 2\omega_{1,1}W_{(\mathrm{dpl})}\kappa + W_{(\mathrm{dpl})} - W_{(\mathrm{dpl})}^2\kappa.$$

Easily, $\mathcal{E}$ is maximized when $\omega_{1,1} = \omega_{2,1} = \frac{1}{2}W_{(\mathrm{dpl})}$, which yields the optimal value $\mathcal{E} = W_{(\mathrm{ttl})} - \frac{1}{2}W_{(\mathrm{dpl})}^2\kappa$. (We did not need to *assume* that the optimal schedule is symmetric in this case; we proved that it should be.)

The principle enunciated at the beginning of this proof implies that an optimal schedule that deploys *overlapping* workloads never allocates a full X units of work to either remote computer. We can, therefore, simplify our calculations by assuming henceforth that $W_{(\mathrm{ttl})} < 2X$. Since we consider only symmetric schedules, Theorem 5 tells us that the common size s of the allocations to the computers satisfies $s \geq W_{(\mathrm{ttl})}/2$. Therefore, the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ as a function of s is

$$\mathcal{E}(s) \;=\; -2s^3\kappa^2 + (2 + W_{(\mathrm{ttl})}\kappa)s^2\kappa - 2sW_{(\mathrm{ttl})}\kappa + W_{(\mathrm{ttl})}.$$

We seek the maximizing value of s.

$$\mathcal{E}'(s) \;=\; \frac{\mathrm{d}}{\mathrm{d}s}\mathcal{E}(s) \;=\; 2[-3s^2\kappa + (2 + W_{(\mathrm{ttl})}\kappa)s - W_{(\mathrm{ttl})}]\kappa.$$

The discriminant of the bracketed quadratic polynomial is:

$$\begin{aligned}\Delta \;&=\; W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4\\ &=\; \left(W_{(\mathrm{ttl})}\kappa - 2\left(2+\sqrt{3}\right)\right)\left(W_{(\mathrm{ttl})}\kappa - 2\left(2-\sqrt{3}\right)\right).\end{aligned}$$

Because $W_{(\mathrm{ttl})} < 2X$ we have, $W_{(\mathrm{ttl})}\kappa < 2(2+\sqrt{3})$. We branch on the relative sizes of $W_{(\mathrm{ttl})}$ and $2(2-\sqrt{3})X$:

$W_{(\mathrm{ttl})} > 2(2-\sqrt{3})X$. In this case, $\Delta < 0$, so the polynomial has no real roots, and $\mathcal{E}(s)$ is decreasing with s. Because $s \in [\frac{1}{2}W_{(\mathrm{ttl})}, W_{(\mathrm{ttl})}]$, $\mathcal{E}(s)$ is maximized when $s = W_{(\mathrm{ttl})}/2$.

$W_{(\mathrm{ttl})} \leq 2(2-\sqrt{3})X$. This case is far more complicated than its predecessor. Let us denote the two roots of our quadratic polynomial by s_- and s_+, as follows:

$$s_- \;=\; \frac{2 + W_{(\mathrm{ttl})}\kappa - \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa} \quad \text{and}$$

$$s_+ \;=\; \frac{2 + W_{(\mathrm{ttl})}\kappa + \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa}.$$

One sees that $\mathcal{E}(s)$ decreases as s progresses from $-\infty$ to s_-, then increases as s progresses from s_- to s_+, then decreases once more as s increases beyond s_+. We must determine how these three intervals overlap $\mathcal{E}$'s domain of validity, viz., $s \in [\frac{1}{2}W_{(\mathrm{ttl})},\ W_{(\mathrm{ttl})}]$. We note first that $\frac{1}{2}W_{(\mathrm{ttl})} \leq s_-$. Indeed:

$$\begin{aligned}& W_{(\mathrm{ttl})}/2 \geq s_-\\ \Leftrightarrow\quad & W_{(\mathrm{ttl})}/2 \geq \frac{2 + W_{(\mathrm{ttl})}\kappa - \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa}\\ \Leftrightarrow\quad & \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4} \geq 2(1 - W_{(\mathrm{ttl})}\kappa)\\ \Rightarrow\quad & 0 \geq 3W_{(\mathrm{ttl})}^2\kappa^2.\end{aligned}$$

We invoke here the fact that $W_{(\mathrm{ttl})}\kappa \leq 1$, because $W_{(\mathrm{ttl})} \leq 2(2-\sqrt{3})X \leq X$. We remark next that $\mathcal{E}'(W_{(\mathrm{ttl})}) = 2W_{(\mathrm{ttl})}\kappa(1 - 2W_{(\mathrm{ttl})}\kappa)$, so that $\mathcal{E}'(W_{(\mathrm{ttl})}) \geq 0$ and $W_{(\mathrm{ttl})} \in [s_-, s_+]$ when $W_{(\mathrm{ttl})} \leq \frac{X}{2}$; moreover, $W_{(\mathrm{ttl})} > s_+$ when $W_{(\mathrm{ttl})} > \frac{1}{2}X$. Indeed, if we assume that $\frac{1}{2}X < W_{(\mathrm{ttl})} \leq s_+$ (the lower bound implying $5W_{(\mathrm{ttl})}\kappa - 2 \geq 0$), then we reach a contradiction:

$$\begin{aligned}& W_{(\mathrm{ttl})} \;\leq\; s_+\\ \Leftrightarrow\quad & W_{(\mathrm{ttl})} \;\leq\; \frac{2 + W_{(\mathrm{ttl})}\kappa + \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa}\\ \Leftrightarrow\quad & 5W_{(\mathrm{ttl})}\kappa - 2 \;\leq\; \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}\\ \Rightarrow\quad & 2W_{(\mathrm{ttl})}\kappa \;\leq\; 1.\end{aligned}$$

So, once again we have two cases to consider:

$W_{(\mathrm{ttl})} \leq X/2$. In this case, we have $W_{(\mathrm{ttl})} \in [s_-, s_+]$, so that $\mathcal{E}(s)$ achieves its maximum either when $s = \frac{1}{2}W_{(\mathrm{ttl})}$ or when $s = W_{(\mathrm{ttl})}$. Hence $\mathcal{E}(s)$'s maximum is

$$\begin{aligned}\text{either}\quad & \mathcal{E}(W_{(\mathrm{ttl})}/2) \;=\; W_{(\mathrm{ttl})} - \tfrac{1}{2}W_{(\mathrm{ttl})}^2\kappa\\ \text{or}\quad & \mathcal{E}(W_{(\mathrm{ttl})}) \;=\; W_{(\mathrm{ttl})} - W_{(\mathrm{ttl})}^3\kappa^2.\end{aligned}$$

When $W_{(\mathrm{ttl})} \leq \frac{1}{2}X$, which is the case here, the latter value dominates, so the optimal deployment is $\omega_{1,1} = \omega_{2,1} = W_{(\mathrm{ttl})}$.

$W_{(\mathrm{ttl})} > X/2$. In this case, $W_{(\mathrm{ttl})} > s_+$, so that $\mathcal{E}(s)$ achieves its maximum either when $s = W_{(\mathrm{ttl})}/2$ or when $s = s_+$. We compare the values at these points by computing both $\mathcal{E}(s_+)$ and $\mathcal{E}(s_+) - \mathcal{E}(W_{(\mathrm{ttl})}/2)$. We find that

$$\begin{aligned}54\kappa\ \mathcal{E}(s_+) \;=\;& (W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4)^{\frac{3}{2}}\\ &+W_{(\mathrm{ttl})}^3\kappa^3 - 12W_{(\mathrm{ttl})}^2\kappa^2 + 30W_{(\mathrm{ttl})}\kappa + 8\end{aligned}$$

and

$$\begin{aligned}54\kappa\ \left(\mathcal{E}(s_+) - \mathcal{E}(W_{(\mathrm{ttl})}/2)\right) \;=\;& \\ &[W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4]^{3/2}\\ &+ [W_{(\mathrm{ttl})}^3\kappa^3 + 15W_{(\mathrm{ttl})}^2\kappa^2 - 24W_{(\mathrm{ttl})}\kappa + 8]. \quad (17)\end{aligned}$$

Easily, both of the bracketed polynomials in (17) *decrease* as $W_{(\mathrm{ttl})}$ progresses along its current hypothesized interval, from $\frac{1}{2}X$ through $2(2-\sqrt{3})X$; therefore, the difference $\mathcal{E}(s_+) - \mathcal{E}(W_{(\mathrm{ttl})}/2)$ decreases as $W_{(\mathrm{ttl})}$ proceeds along the same interval. Since the difference vanishes at the point $W_{(\mathrm{ttl})} = \frac{1}{2}X$, we conclude that the optimal deployment in this case is $\omega_{1,1} = \omega_{2,1} = \frac{1}{2}W_{(\mathrm{ttl})}$. □

4.3.2 (Asymptotic) optimality for multi-chunk allocations

This section is devoted to presenting and analyzing Algorithm 1, which prescribes schedules for two remote computers that are always asymptotically optimal and that are exactly optimal when $W_{(\mathrm{ttl})} \geq 2X$. We use the notation Algorithm $1(n)$ to denote the n-chunk invocation of the algorithm. The algorithm employs the notation $\langle a, b\rangle$ for sub-workloads, that we introduced at the beginning of Section 3.1.2.

Theorem 8: ((Asymptotically) optimal schedule: linear risk) The schedules specified by Algorithm 1 have the following performance.

1) When $W_{(\mathrm{ttl})} \geq 2X$, they are *exactly optimal*, completing, in expectation, $(1 - 1/n)X$ units of work.

Algorithm 1. Scheduling for 2 computers using at most n chunks per computer; n is an input.

if $W_{(\mathrm{ttl})} \geq 2X$ **then**

$\quad \forall i \in [1,n]\ \ \mathcal{W}_{1,i} \leftarrow \left\langle \frac{i-1}{n} \cdot \frac{n}{n+1}X,\ \frac{i}{n} \cdot \frac{n}{n+1}X \right\rangle$

$\quad \forall i \in [1,n]\ \ \mathcal{W}_{2,i} \leftarrow \left\langle W_{(\mathrm{ttl})} - \frac{i}{n} \cdot \frac{n}{n+1}X,\ W_{(\mathrm{ttl})} - \frac{i-1}{n} \cdot \frac{n}{n+1}X \right\rangle$

if $W_{(\mathrm{ttl})} \leq X$ **then**

$\quad \forall i \in [1,n]\ \ \mathcal{W}_{1,i} = \mathcal{W}_{2,n-i+1} \leftarrow \left\langle \frac{i-1}{n}W_{(\mathrm{ttl})},\ \frac{i}{n}W_{(\mathrm{ttl})} \right\rangle$

if $X < W_{(\mathrm{ttl})} < 2X$ **then**

$\quad \ell \leftarrow \lfloor n/3 \rfloor$

$\quad \forall i \in [1,\ell]\ \ \mathcal{W}_{1,i} \leftarrow \left\langle \frac{i-1}{\ell}(W_{(\mathrm{ttl})} - X),\ \frac{i}{\ell}(W_{(\mathrm{ttl})} - X) \right\rangle$

$\quad \forall i \in [1,\ell]\ \ \mathcal{W}_{2,i} \leftarrow \left\langle W_{(\mathrm{ttl})} - \frac{i}{\ell}(W_{(\mathrm{ttl})} - X),\ W_{(\mathrm{ttl})} - \frac{i-1}{\ell}(W_{(\mathrm{ttl})} - X) \right\rangle$

$\quad \forall i \in [1,2\ell]\ \ \mathcal{W}_{1,\ell+i} = \mathcal{W}_{2,3\ell-i+1} \leftarrow \left\langle (W_{(\mathrm{ttl})} - X) + \frac{i-1}{2\ell}(2X - W_{(\mathrm{ttl})}),\ (W_{(\mathrm{ttl})} - X) + \frac{i}{2\ell}(2X - W_{(\mathrm{ttl})}) \right\rangle$

2) When $W_{(\mathrm{ttl})} < 2X$, they are *asymptotically optimal*.
   a) When $W_{(\mathrm{ttl})} \leq X$, their expected work production tends asymptotically to $W_{(\mathrm{ttl})} - \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2$ units.
   b) When $X < W_{(\mathrm{ttl})} < 2X$, their expected work production tends asymptotically to $2W_{(\mathrm{ttl})} - \frac{1}{3}X - W_{(\mathrm{ttl})}^2\kappa + \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2$ units.

Note. The actual (nonasymptotic) work productions from parts 2(a) and 2(b) are, respectively,

- For part 2(a): $W_{(\mathrm{ttl})} - \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2\left(1 + \frac{3}{n} + \frac{2}{n^2}\right)$ work units.
- For part 2(b):

$$2W_{(\mathrm{ttl})} - \tfrac{1}{3}X - W_{(\mathrm{ttl})}^2\kappa + \tfrac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2 + \tfrac{1}{m}\left(\left(1+\tfrac{1}{m}\right)W_{(\mathrm{ttl})} - \left(1+\tfrac{2}{3m}\right)X - \tfrac{1}{2m}W_{(\mathrm{ttl})}^2\kappa - \tfrac{1}{4}\left(1-\tfrac{1}{3m}\right)W_{(\mathrm{ttl})}^3\kappa^2\right)$$

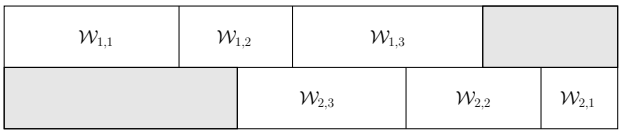
work units, where $m = \lfloor n/3 \rfloor$.

Proof: Case 1: $W_{(\mathrm{ttl})} \geq 2X$. By definition, a remote computer is certain to be interrupted when its subworkload has size $\geq X$. Therefore, Theorem 5 tells us that when $W_{(\mathrm{ttl})} \geq 2X$, we lose no work production by insisting that the two computers work on disjoint subsets of the workload. Theorem 2 tells us how to schedule the subworkloads, and it gives the expectation of $\mathsf{W}_{(\mathrm{cmp})}$.

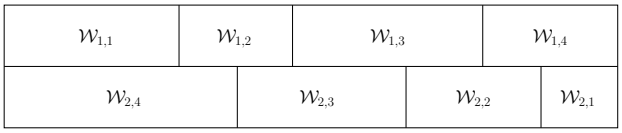
Case 2: $W_{(\mathrm{ttl})} < 2X$. This case requires a subtler argument. Focus on a fixed but arbitrary integer $n > 0$. Clearly, the expected work completed by Algorithm 1 when each remote computer's workload is partitioned into n chunks cannot exceed the analogous quantity for the optimal n-chunk schedule $\Sigma(n)$. Note that we lose no generality by insisting that schedule $\Sigma(n)$ enjoys the three characteristics of Theorem 5.

Subcase a: $W_{(\mathrm{ttl})} < X$. We depict schedule $\Sigma(n)$ in a convenient schematic form in Fig. 2(a).

Strategy. We show that, with the current bound on $W_{(\mathrm{ttl})}$, $\Sigma(n)$'s expected work production is no greater than that of the $(2n+1)$-chunk schedule produced by Algorithm 1. We then derive the value of the expected work production of our schedules, which yields Case 2(a) of the theorem. Our analysis builds on three successive work-preserving transformations to schedule $\Sigma(n)$,

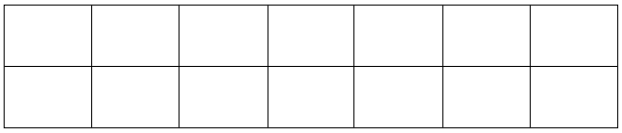


(a) Optimal n-chunk schedule $\Sigma(n)$.



(b) $\Sigma_1(n)$: each computer gets an $(n+1)$th chunk so that it processes all of $W_{(\mathrm{ttl})}$.

(c) $\Sigma_2(n)$: $\Sigma_1(n)$ with chunks subdivided to make chunk boundaries coincide.



(d) $\Sigma_3(n)$: $\Sigma_2(n)$ implemented via Algorithm 1 with $2n+1$ chunks.

Fig. 2. Schedule transformations used to prove asymptotic optimality results in Theorems 6 and 8 (Algorithm 1 when $W_{(\mathrm{ttl})} \leq X$).

that are depicted schematically in Fig. 2.

Transformations. (1) We transform $\Sigma(n)$ to a schedule $\Sigma_1(n)$ that has each computer process the entire workload $W_{(\mathrm{ttl})}$, by adding a possibly empty $(n+1)$th chunk to the workload of each computer; see Fig. 2(b). (2) We transform $\Sigma_1(n)$ to a schedule $\Sigma_2(n)$ within which both computers' chunk boundaries coincide; see Fig. 2(c). We accomplish this as follows. Focus on the $(n+1)$-element increasing sequences, $\vec{\mathcal{B}}_1$ and $\vec{\mathcal{B}}_2$, of "places" in the workloads of computers P_1 and P_2, respectively, at which the chunks end:

$$\begin{aligned}\vec{\mathcal{B}}_1 &= \omega_{1,1},\ (\omega_{1,1}+\omega_{1,2}),\ldots,\ \sum_{j=1}^{n+1}\omega_{1,j}\\ \vec{\mathcal{B}}_2 &= W_{(\mathrm{ttl})}-\sum_{j=1}^{n+1}\omega_{2,j},\ W_{(\mathrm{ttl})}-\sum_{j=1}^{n}\omega_{2,j},\\ &\quad \ldots, W_{(\mathrm{ttl})}-(\omega_{2,1}+\omega_{2,2}),\ W_{(\mathrm{ttl})}-\omega_{2,1}\end{aligned}$$

Merge the elements of $\vec{\mathcal{B}}_1$ and $\vec{\mathcal{B}}_2$ into an increasing sequence, $b_0 < b_1 < \cdots < b_l = W_{(\mathrm{ttl})}$; clearly, $l \le 2n+1$, because $\vec{\mathcal{B}}_1$ and $\vec{\mathcal{B}}_2$ could contain many elements in common and *must* both end with $W_{(\mathrm{ttl})}$. Then specify the new, coterminal, chunks by partitioning $W_{(\mathrm{ttl})}$ into l chunks as follows:

$$\mathcal{W}'_{1,i} = \mathcal{W}'_{2,l-i+1},\ \text{where each } \omega'_{1,i} = b_i - b_{i-1}.$$

(3) We invoke Algorithm $1(l)$ to produce schedule $\Sigma_3(n)$; note that we now have l *equal-size* chunks; see Fig. 2(d):

$$(\forall i \in [1,l])\ \ \mathcal{W}''_{1,i} = \mathcal{W}''_{2,l-i+1} = \left[\frac{i-1}{l}W_{(\mathrm{ttl})},\ \frac{i}{l}W_{(\mathrm{ttl})}\right).$$

(4) We finally replace $\Sigma_3(n)$ by the schedule $\widehat{\Sigma}(n)$ produced by invoking Algorithm $1(2n+1)$, thereby increasing l to its maximum value.

Analysis. (1) Our failure model ensures that schedule $\Sigma_1(n)$'s expected work production is no smaller than $\Sigma(n)$'s. (2) Because we focus here only on the free-initiation model, subdividing chunks cannot decrease the overall expected work production. Hence, $\Sigma_2(n)$'s expected work production is no smaller than $\Sigma_1(n)$'s. (3) The only problematic transformation is the one that replaces $\Sigma_2(n)$ by $\Sigma_3(n)$. It is not obvious that this alteration cannot decrease work production. We prove this via the following more general result. Consider the following schedule-optimization problem for computers, P_1 and P_2. For $i = 1, 2$, computer P_i executes l_i chunks of possibly different sizes, $\mathcal{V}_{i,1}$, …, $\mathcal{V}_{i,l_i}$, in that order. Suppose that P_1 and P_2 have two consecutive chunks in common; i.e., for some $i \in [1, l_1 - 1]$ and $j \in [1, l_2 - 1]$, $\mathcal{V}_{1,i} = \mathcal{V}_{2,j+1}$ and $\mathcal{V}_{1,i+1} = \mathcal{V}_{2,j}$.[7] What should the relative sizes of $\mathcal{V}_{1,i}$ (=$\mathcal{V}_{2,j+1}$) and $\mathcal{V}_{1,i+1}$ (=$\mathcal{V}_{2,j}$) be? We begin to answer this question by considering the impact on the overall work expectation $\mathcal{E}$ of possibly redistributing work between chunks $\mathcal{V}_{1,i}$ and $\mathcal{V}_{1,i+1}$. To simplify formulas, let $V_1 = \sum_{k=1}^{i-1}|\mathcal{V}_{1,k}|$, $V_2 = \sum_{k=1}^{j-1}|\mathcal{V}_{2,k}|$, and $L = |\mathcal{V}_{1,i}| + |\mathcal{V}_{1,i+1}|$. Then:

$$\begin{aligned}\mathcal{E} &= |\mathcal{V}_{1,i}|\,(1-(V_1+|\mathcal{V}_{1,i}|)\kappa(V_2+L)\kappa)\\ &\quad +(L-|\mathcal{V}_{1,i}|)\,(1-(V_1+L)\kappa(V_2+L-|\mathcal{V}_{1,i}|)\kappa)\,.\end{aligned}$$

Thus, the contribution of these two chunks to $\mathcal{E}$ is:

$$\begin{aligned}\mathcal{E} &= (V_1+V_2+2L)L\kappa^2|\mathcal{V}_{1,i}|\\ &\quad +L(1-(V_1+L)(V_2+L)\kappa^2)\\ &\quad -(V_1+V_2+2L)\kappa^2|\mathcal{V}_{1,i}|^2,\end{aligned}$$

which is maximized by setting $|\mathcal{V}_{1,i}| = \frac{1}{2}L$, i.e., by making both chunks the same size.

Proceeding by induction, we generalize from the case $l = 2$, to show that replacing l consecutive chunks by l equal-size chunks cannot decrease work expectation. (4) We note finally that $\widehat{\Sigma}(n)$'s expected work production is no smaller than $\Sigma_3(n)$'s. This follows from the next paragraph, in which we show that the expected work production of schedules produced via Algorithm 1 increases with the number n of chunks; cf. (18).

The endgame of the analysis. We obtain the desired bound on the performance of Algorithm 1. Let $E(W_{(\mathrm{ttl})}, \Sigma(n))$ and $E(W_{(\mathrm{ttl})}, \mathrm{A}(n))$ denote, respectively, the work expectation of the optimal schedule and Algorithm 1's schedule, when each deploys n equal-size chunks per computer. We find

$$\begin{aligned}&E(W_{(\mathrm{ttl})}, \mathrm{A}(n))\\ &\quad= \sum_{i=1}^{n}\frac{W_{(\mathrm{ttl})}}{n}\left(1-\sum_{j=1}^{i}\frac{W_{(\mathrm{ttl})}}{n}\kappa\cdot\sum_{j=1}^{n-i+1}\frac{W_{(\mathrm{ttl})}}{n}\kappa\right)\\ &\quad= W_{(\mathrm{ttl})}-\frac{W^3_{(\mathrm{ttl})}\kappa^2}{6}\left(1+\frac{3}{n}+\frac{2}{n^2}\right),\end{aligned}\tag{18}$$

which is obviously nondecreasing in n.

We have thus proved that, for all $n > 0$,

$$E(W_{(\mathrm{ttl})}, \mathrm{A}(2n+1)) \ge E(W_{(\mathrm{ttl})}, \Sigma(n)) \ge E(W_{(\mathrm{ttl})}, \mathrm{A}(n)).$$

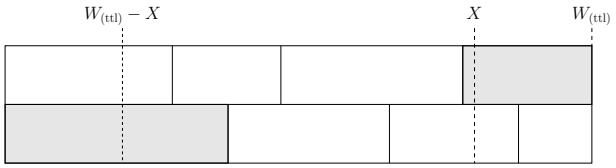
$E(W_{(\mathrm{ttl})}, \Sigma(n))$ is obviously a nondecreasing function bounded above by $W_{(\mathrm{ttl})}$; hence, it has a limit as n increases without bound. The preceding inequalities show that $E(W_{(\mathrm{ttl})}, \Sigma(n))$ and $E(W_{(\mathrm{ttl})}, \mathrm{A}(n))$ have the same limit, whence Algorithm 1's schedule is asymptotically optimal. By (18) the limit shared by $E(W_{(\mathrm{ttl})}, \Sigma(n))$ and $E^{(\mathrm{A})}(n)$ is $W_{(\mathrm{ttl})} - \frac{1}{6}W^3_{(\mathrm{ttl})}\kappa^2$.

Subcase b: $X < W_{(\mathrm{ttl})} < 2X$. We depict schedule $\Sigma(n)$ in a convenient schematic form in Fig. 3(a).

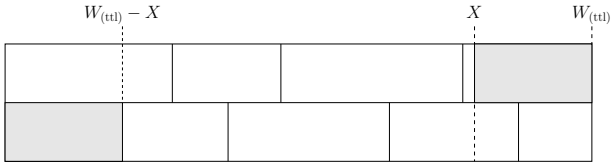
Strategy. We parallel the argument of Case 2(a), with two changes. We now craft *four* work-preserving transformations to $\Sigma(n)$ (depicted schematically in Fig. 3), and we now show that, with the current bounds on $W_{(\mathrm{ttl})}$, $\Sigma(n)$'s expected work production is no greater than that of the $3(n+1)$-chunk schedule produced by Algorithm 1.

Transformations. (1) We replace $\Sigma(n)$ by schedule $\Sigma_1(n)$, that deploys exactly X units of work to each remote computer, by adding an $(n+1)$th chunk to the workload of each computer; see Fig. 3(b).

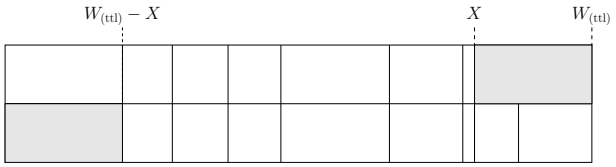
7. Theorem 5 tells us that P_1 and P_2 should execute these chunks in opposite orders.



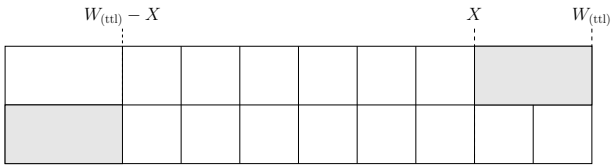
(a) $\Sigma(n)$: Optimal schedule with n chunks.



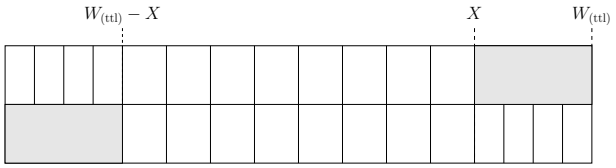
(b) $\Sigma_1(n)$: Extending $\Sigma(n)$'s subworkloads to size X via an $(n+1)$th chunk.



(c) $\Sigma_2(n)$: Subdividing chunks to line up boundaries.



(d) $\Sigma_3(n)$: Equalizing chunk sizes in each of the three intervals.



(e) $\Sigma_4(n)$: Algorithm 1's schedule with $3(n+1)$ chunks.

Fig. 3. Schedule transformations that prove the asymptotic optimality of Algorithm 1 when $X < W_{(\mathrm{ttl})} < 2X$.

(2) Proceeding as in Subcase a(2), we produce schedule $\Sigma_2(n)$, that subdivides the chunks of $\Sigma_1(n)$ when necessary in order to align the chunk boundaries of both computers; see Fig. 3(c).

The work charts depicted in Fig. 3(b)–3(e) consist of three intervals, whose boundaries are also chunk boundaries: $I^{(\mathrm{left})} = [0, W_{(\mathrm{ttl})} - X]$, $I^{(\mathrm{mid})} = [W_{(\mathrm{ttl})} - X, X]$, and $I^{(\mathrm{right})} = [X, W_{(\mathrm{ttl})}]$. The chunk boundaries in $I^{(\mathrm{left})}$ come only from original chunks of P_1 and from the $(n+1)$th chunk that we added to P_2; $I^{(\mathrm{left})}$ thus contains $\leq n+1$ chunks. Easily, the same bound holds for $I^{(\mathrm{right})}$, with the computers switching roles. Turning to $I^{(\mathrm{mid})}$, note that all the boundaries of chunks $\mathcal{W}_{1,2}, \ldots, \mathcal{W}_{1,n}$, and of chunks $\mathcal{W}_{2,2}, \ldots, \mathcal{W}_{2,n}$, could lie strictly within this interval; therefore, in the worst case, $2n$ chunk boundaries could lie strictly between $W_{(\mathrm{ttl})} - X$ and X; hence, $I^{(\mathrm{mid})}$ could contain as many as $2n+1$ chunks.

(3) We replace $\Sigma_2(n)$ by schedule $\Sigma_3(n)$, that equalizes the sizes of chunks independently in each of $I^{(\mathrm{left})}$, $I^{(\mathrm{mid})}$, and $I^{(\mathrm{right})}$; see Fig. 3(d).

(4) Finally, we produce schedule $\Sigma_4(n)$ by using Algorithm $1(3(n+1))$ to build a schedule with $n+1$ chunks in interval $I^{(\mathrm{left})}$, $2n+2$ chunks in interval $I^{(\mathrm{mid})}$, and $n+1$ chunks in interval $I^{(\mathrm{right})}$. Note that $\Sigma_4(n)$ employs at least as many chunks in each interval as $\Sigma_3(n)$ does and that all chunks in each interval are like-sized; see Fig. 3(e).

Analysis. (1) By definition of X, $\Sigma_1(n)$ has the same work expectation as $\Sigma(n)$. (2,4) We noted in Subcase a that neither lining up chunk boundaries nor increasing the number of chunks can decrease the work expectation of either computer. (3) Once again, the problematic transformation is the one that equalizes chunk sizes interval by interval: it is not clear that this transformation cannot decrease the overall work expectation. Now, it *is* clear from the proof of Case 1 that this transformation does not decrease the work expectation when it is applied solely to the interval $I^{(\mathrm{mid})}$. Therefore, one needs focus only on the two "outer" intervals. This task is simplified under our interruption/failure model, because the disjointness of intervals $I^{(\mathrm{left})}$ and $I^{(\mathrm{right})}$ allows us to analyze them independently of one another. Therefore, we lose no generality by focusing solely on interval $I^{(\mathrm{left})}$. To this end, we consider two consecutive chunks of P_1, $\mathcal{V}_{1,i}$ and $\mathcal{V}_{1,i+1}$, that both belong to interval $I^{(\mathrm{left})}$ (hence, do not intersect the workload of P_2). Letting $L = |\mathcal{V}_{1,i}| + |\mathcal{V}_{1,i+1}|$ and $V = \sum_{j=1}^{i-1} \mathcal{V}_{1,j}$, we find that these chunks contribute

$$\mathcal{E} \;=\; \left(|\mathcal{V}_{1,i}|\kappa + (1 - (V+L)\kappa)\right) L - |\mathcal{V}_{1,i}|^2\kappa$$

units of work to the overall expectation. This expression is obviously maximized when $|\mathcal{V}_{1,i}| = L/2$, i.e., when the two consecutive chunks have the same size. (4) The last transformation increases the numbers of same-size jobs in each of the three intervals to their maximum possible respective values: $n+1$, $2(n+1)$, and $n+1$ (see Figure 3(e)). Direct calculation, coupled with the analysis of subcase 2(a) shows that this cannot decrease expected work production for the chunks in interval $I^{(\mathrm{mid})}$. We now show that this is also the case for chunks in interval $I^{(\mathrm{left})}$. To wit, the cumulative expectation for the m equal-size chunks of $I^{(\mathrm{left})}$ is:

$$\begin{aligned}\mathcal{E} \;&=\; \sum_{i=1}^{m} \frac{W_{(\mathrm{ttl})} - X}{m} \left(1 - \sum_{j=1}^{i} \frac{W_{(\mathrm{ttl})} - X}{m}\kappa\right) \\ &=\; (W_{(\mathrm{ttl})} - X) - \left(\frac{1}{2} + \frac{1}{2m}\right)(W_{(\mathrm{ttl})} - X)^2\kappa,\end{aligned}$$

which is obviously increasing in m.

The expectation of $\mathsf{W}_{(\mathrm{cmp})}$, with $l = \lfloor \frac{n}{3} \rfloor$, for the scheduling of Algorithm 1(n) is then equal to:

$$\begin{aligned}
\mathcal{E} = & \sum_{i=1}^{l} \frac{W_{(\mathrm{ttl})} - X}{l} \left(1 - i\frac{W_{(\mathrm{ttl})} - X}{l}\kappa\right) \\
& + \sum_{i=1}^{2l} \frac{2X - W_{(\mathrm{ttl})}}{2l} \left(1 - \left(W_{(\mathrm{ttl})} - X + i\frac{2X - W_{(\mathrm{ttl})}}{2l}\right)\kappa \right. \\
& \qquad \left. \times \left(X - (i-1)\frac{2X - W_{(\mathrm{ttl})}}{2l}\right)\kappa\right) \\
& + \sum_{i=1}^{l} \frac{W_{(\mathrm{ttl})} - X}{l} \left(1 - i\frac{W_{(\mathrm{ttl})} - X}{l}\kappa\right) \\
= & \quad 2W_{(\mathrm{ttl})} - \frac{1}{3}X - W_{(\mathrm{ttl})}^2\kappa + \frac{W_{(\mathrm{ttl})}^3\kappa^2}{6} \\
& + \frac{1}{l}\left(\left(1 + \frac{1}{l}\right)W_{(\mathrm{ttl})} - \left(1 + \frac{2}{3l}\right)X \right. \\
& \qquad \left. - \frac{1}{2l}W_{(\mathrm{ttl})}^2\kappa - \frac{1}{4}\left(1 - \frac{1}{3l}\right)W_{(\mathrm{ttl})}^3\kappa^2\right).
\end{aligned}$$

□

5 Strategies for Crafting General Schedules

This section is devoted to developing strategies that allow one to craft simple heuristics for sharing work with arbitrary numbers of remote computers that are subject to unrecoverable interruptions. In an attempt to describe our heuristics in terms that are clear, evocative, and free of undesired connotations, we introduce the following terminology.

- A *coterie* is a collection of remote computers that work jointly on the same subset of our workload.
- A *slice* of our workload is a subset that we assign to a single coterie.

All of our scheduling heuristics operate as follows.

1) They *partition* the total workload into slices that they deploy on disjoint coteries.
2) They tell each coterie how to partition its assigned slices into *equal-size* chunks.
3) They orchestrate the processing of the slices on each coterie.

The entire scheduling process is performed in the light of our interruption model. Specifically, we acknowledge the futility of deploying a slice of work of size $> X$ on any remote computer, because such a slice is "certain" to be interrupted—i.e., will be interrupted with probability 1. The amount of work that we actually deploy to the remote computers, which we denote by $W_{(\mathrm{dpl})}$, is, therefore, often less than the entire $W_{(\mathrm{ttl})}$ units of work in the overall workload. Part of the scheduling process is to determine the size of $W_{(\mathrm{dpl})}$.

5.1 The Partitioning Phase

We begin the scheduling process with some simple heuristics that determine the value of $W_{(\mathrm{dpl})}$ and that partition this many units of work into slices that will be deployed on the remote computers. While these heuristics are tailored to the linear risk function, we show how to adapt them easily to other risk functions. Our scheduling strategy branches into three sub-strategies that depend heavily on the size of the overall workload $W_{(\mathrm{ttl})}$.

$W_{(\mathrm{ttl})}$ is "very large." When $W_{(\mathrm{ttl})} \geq pX$, we deploy p slices, each of size X, to be processed independently on the remote computers. We abandon the remaining $W - pX$ units of work, because our interruption model tells us that it is "certain" not to be completed. (We assume that work is not prioritized, so we do not care *which* pX units we deploy.) We schedule the deployed work on each remote computer in the manner prescribed by the single-computer scheduling guidelines in Section 3.

$W_{(\mathrm{ttl})}$ is "very small." When $W_{(\mathrm{ttl})} \leq X$, we deploy the entire workload in a single slice, which we replicate on all p remote computers. We have thus a single coterie with p computers which works on a slice of size X, and we orchestrate the work as explained in Section 5.2.

$W_{(\mathrm{ttl})}$ is of "intermediate" size. The case $X < W_{(\mathrm{ttl})} < pX$ is the interesting challenge, as there is no compelling known scheduling strategy. Here we can deploy the total work $W_{(\mathrm{ttl})}$, and replicate each deployed piece $pX/W_{(\mathrm{ttl})}$ times on average. Note that no worker will receive more than X units of work with this deployment scheme. In our quest for schedules that are simple to implement, we assign disjoint coteries to work on independent slices of work. Ideally, we would like to have $p/(pX/W_{(\mathrm{dpl})}) = W_{(\mathrm{dpl})}\kappa$ coteries, but this number may not be integral, so we approximate this ideal by partitioning the workload into $q = \lceil W_{(\mathrm{dpl})}\kappa \rceil$ slices. We balance computing resources as much as possible, by replicating each slice on either $\lfloor p/q \rfloor$ or $\lceil p/q \rceil$ remote computers. We balance the load among the coteries by having a coterie with $\lfloor p/q \rfloor$ computers (resp., $\lceil p/q \rceil$ computers) work on a slice of size $sl^- = \lfloor p/q \rfloor W_{(\mathrm{dpl})}/p$ (resp., of size $sl^+ = \lceil p/q \rceil W_{(\mathrm{dpl})}/p$).

Adapting to general risk functions. The preceding scenario is tailored to the linear risk function in that the load of a single computer has a size $\leq X$. To adapt the strategy to general risk functions, we would introduce a parameter λ that specifies the maximum probability of interruption that the user would allow for the work allocated to a computer. (For linear risk, we set $\lambda = 1$, a choice that could often be impractical; consider, e.g., a heavy-tailed distribution.) We would then use λ to compute the maximum size of a slice, $maxsl$, by insisting that $Pr(maxsl) = \lambda$. For illustration, if $\lambda = 1/2$, then under the linear risk function we would set $maxsl = \frac{1}{2}X$, while with the exponential risk function we would set $maxsl = (\ln 2)X$. The deployed workload would now consist of $W_{(\mathrm{dpl})} = \min(W_{(\mathrm{ttl})},\ p \times maxsl)$ units—which would mandate using $q = \lceil W_{(\mathrm{dpl})}/maxsl \rceil$ slices, whose sizes are as defined previously.

Computer \ Chunk	1	2	3	4	5	6	7	8	9	10	11	12
P_1	1	6	9	12	2	5	8	11	3	4	7	10
P_2	12	1	6	9	11	2	5	8	10	3	4	7
P_3	9	12	1	6	8	11	2	5	7	10	3	4
P_4	6	9	12	1	5	8	11	2	4	7	10	3

TABLE 1
Execution chart for a general schedule.

Group 1 chunks 1–4	Group 2 chunks 5–8	Group 3 chunks 9–12
1	2	3
6	5	4
9	8	7
12	11	10

TABLE 2
Execution chart for a group schedule.

5.2 The Orchestration Phase

The partitioning phase of Section 5.1 has formed mutually disjoint slices of work, each of size sl, that we deploy to disjoint coteries. We first partition each slice into some number of equal-size chunks: We have shown that such equal-size-chunk schedules are asymptotically optimal when scheduling one or two remote computers (Theorems 3 and 6, respectively). The first issue then to resolve is, what should the *checkpointing granularity* be, as measured by the number of chunks we partition each slice into? We denote this quantity by n, so that each remote computer partitions its slice into n chunks of common size $\omega = sl/n$. For the well-structured heuristics that we design here, we can actually determine the best value of n. We defer this determination until Section 5.2.3, since we have to develop some concepts first. Each chunk of work that is deployed to a coterie Γ will be scheduled on every one of Γ's g computers. Our challenge is to determine the time step at which each chunk i will be scheduled on each computer j of Γ. We strive for a schedule that maximizes the expected amount of work completed by the total assemblage of p computers. We discuss the orchestration phase in detail only for the linear risk model. We generalize this discussion to arbitrary risk functions in Section 7.1.

5.2.1 Overview

We illustrate our approach to orchestration via an example wherein each coterie contains $g = 4$ computers and each slice is partitioned into $n = 12$ chunks. Because coteries operate mutually independently, we need to develop a schedule for just one coterie and replicate it on all of the others. Given a coterie Γ and its associated slice, we represent a possible schedule via an *execution chart C for* Γ, as depicted in Table 1. Each row of chart C represents a computer in Γ, and each column represents one of the chunks into which Γ's slice is chopped. Chart entry $C_{i,j}$ is the step at which chunk j is processed by computer P_i.

Any $g \times n$ integer matrix whose rows are permutations of $[1..n]$ is a valid execution chart, under which each P_i executes each chunk j precisely once, at step $C_{i,j}$. One can use an execution chart to calculate the expected amount of work completed under the schedule Σ that the chart specifies. To wit, chunk j will *not* be executed under the schedule Σ only if all g computers in the coterie are interrupted before they complete the chunk. This occurs with probability $\prod_{i=1}^{g} Pr(C_{i,j}\omega) = \prod_{i=1}^{g} Pr\left(C_{i,j} sl/n\right)$, so the expected amount of work completed from the slice is

$$E(sl, n, \Sigma) \;=\; \left(1 - \frac{1}{n}\left(\frac{sl \cdot \kappa}{n}\right)^{g} \sum_{j=1}^{n} \prod_{i=1}^{g} C_{i,j}\right) sl. \quad (19)$$

This last expression is specific to the linear risk model and assumes that

$$Pr(C_{i,j}\omega) = C_{i,j}\omega\kappa \leq 1. \quad (20)$$

We retain assumption (20) while computing expectations throughout this section.

Note. We retain assumption (20) for pragmatic reasons that are justified by our simulations. Under our partitioning strategy, the assumption is true when a coterie has $\lfloor p/q \rfloor$ computers, because $sl^{-} \leq X$, which ignores cases when $sl^{+} > X$. Taking the latter cases into account would considerably complicate all expectation formulas, thereby preventing us from drawing conclusions when comparing heuristics. It is, therefore, convenient to retain assumption (20) even when we know it does not always hold. Fortunately, the conclusions that we reach in Section 5.3 using this assumption are supported by the experiments, which consider *all* cases (see Section 6 and the appendices). These experiments thus provide an a posteriori justification for the simplifying assumption. In other words, the performance of the coteries with $\lfloor p/q \rfloor$ computers empirically gives us good insight into the actual performance of heuristics.

One can derive the following upper bound on $E(sl, n, \Sigma)$, to lend perspective to that expectation.

Proposition 1: For any schedule Σ,

$$E(sl, n, \Sigma) \leq \left(1 - \left(sl \cdot \kappa \frac{(n!)^{1/n}}{n}\right)^{g}\right) sl \approx \left(1 - \left(\frac{sl \cdot \kappa}{e}\right)^{g}\right) sl.$$

Proof: Let $cp_j = \prod_{i=1}^{g} C_{i,j}$ be the j-th column product in the chart. From the expression of $E(sl, n, \Sigma)$, we see that it is maximum when the sum of the n column products is minimum. But the product of the column products is constant, because each row is a permutation of $[1..n]$: we have $\prod_{j=1}^{n} cp_j = (n!)^g$. The sum is minimum when all products are equal (to $(n!)^{\frac{g}{n}}$), whence the inequality. Stirling's formula gives a useful approximation of the upper bound when n is large:

$$E_{\max} \approx sl \left(1 - \left(\frac{sl \cdot \kappa}{e}\right)^{g}\right).$$

□

5.2.2 Group Schedules: Introduction

One notes that under the schedule of Table 1, chunks 1–4 are always executed at the same steps—but by different computers; the same is true for chunks 5–8 as a group and for chunks 9–12 as a group. The twelve chunks of the slice thus partition naturally into three *groups*. One can, therefore, re-specify the schedule of Table 1 as the *group-(oriented)* schedule of Table 2, thereby significantly simplifying the specification. In the group execution chart of Table 2, each column corresponds to a group of chunks, and the ith row specifies the step at which chunks are executed for the ith time. (Thus, for instance, each chunk in group 1 is executed for the first time at step 1, for the second time at step 6, and so on.) This specification leaves implicit that chunk indices within each group are permuted cyclically at each step, so that each chunk is scheduled on each computer exactly once.

We generalize this description. When n is a multiple of g, we can sometimes convert the full $g \times n$ execution chart C exemplified by Table 1 to the $g \times n/g$ group execution chart $\widehat{C}$ exemplified by Table 2. There are n/g groups of size g, and chart-entry $\widehat{C}_{i,j}$ denotes the step at which group j of chunks is executed for the ith time. For a chart $\widehat{C}$ to specify a valid group schedule Σ, its total set of entries must be a permutation of $[1..n]$. In this case, we denote the chart by $\widehat{C}^{(\Sigma)}$. We can compute easily the expected amount of work that schedule Σ completes under the linear risk model:

$$E(sl, n, \Sigma) \;=\; sl - \mathsf{K}^{(\Sigma)} \frac{g}{\kappa} \left(\frac{sl \cdot \kappa}{n}\right)^{g+1}$$

where Σ's *performance constant* $\mathsf{K}^{(\Sigma)}$ is

$$\mathsf{K}^{(\Sigma)} \;=\; \sum_{j=1}^{n/g} \prod_{i=1}^{g} \widehat{C}_{i,j}^{(\Sigma)}.$$

Note that *a smaller value of* $\mathsf{K}^{(\Sigma)}$ *corresponds to a larger value of* $E(sl, n, \Sigma)$.

Group schedules are very natural, because they are *symmetric:* all computers are allocated the same number of chunks, and each computer's schedule is a "translate" of each other computer's; i.e., each computer is scheduled to execute a chunk from the ith group at the same steps as every other computer. Intuition suggests that the most productive schedules are symmetric: Why should some of the identical computers be treated differently from others by a "nature" that interrupts all computers at random times (within the distribution specified by the risk function)? Lending perspective to (19) and confidence in the upper bound of Proposition 1—and, more generally, in our focus on group schedules—is the fact that we have not been able to strengthen the latter bound for group schedules! In fact, the latter bound affords us an easy lower bound on the performance constant of any group schedule that has parameters g and n: $\mathsf{K}_{\min} = \lceil (n/g)(n!)^{g/n} \rceil$. We shall see in Section 5.2.4 that this value of $\mathsf{K}_{\min}$ cannot be improved in general.

5.2.3 Choosing the Granularity of Checkpointing

We are finally ready to determine the best number n of chunks to partition each computer's workslice into. Happily, at least for group schedules, one does not have to guess at this value. We begin to flesh out this remark by noting that we can easily obtain an explicit expression for the expected work completed by any group schedule under the *charged-initiation model*, from that schedule's analogous expectation under the free-initiation model. Moreover, this expression is valid for any nondecreasing risk function.

Theorem 9: Let Σ be a group schedule defined by the execution chart $C_{i,j}\,\big|_{i\in\{1,\ldots,g\},j\in\{1,\ldots,n/g\}}$. Then, whatever the (nondecreasing) risk function,

$$E^{(c,n)}(sl^{(c)}, \Sigma) \;=\; \frac{sl^{(c)}}{sl^{(c)} + n\varepsilon} E^{(f,n)}(sl^{(c)} + n\varepsilon, \Sigma).$$

Proof: To establish the result, we only need to explicit the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ under the charged-initiation model:

$$\begin{aligned}
&E^{(c,n)}(sl^{(c)}, \sigma) \\
&\quad= \sum_{j=1}^{n} \frac{sl^{(c)}}{n} \left(1 - \prod_{i=1}^{g} Pr^{(c)}\left(C_{i,j} \frac{sl^{(c)}}{n}\right)\right) \\
&\quad= \sum_{j=1}^{n} \frac{sl^{(c)}}{n} \left(1 - \prod_{i=1}^{g} Pr^{(f)}\left(C_{i,j} \left(\frac{sl^{(c)}}{n} + \varepsilon\right)\right)\right) \\
&\quad= \frac{sl^{(c)}}{sl^{(c)}+n\varepsilon} \sum_{j=1}^{n} \left(\frac{sl^{(c)}}{n} + \varepsilon\right)\left(1 - \prod_{i=1}^{g} Pr^{(f)}\left(C_{i,j} \left(\frac{sl^{(c)}}{n} + \varepsilon\right)\right)\right) \\
&\quad= \frac{sl^{(c)}}{sl^{(c)} + n\varepsilon} E^{(f,n)}(sl^{(c)} + n\varepsilon, \sigma).
\end{aligned}$$

□

We can now determine the optimal value for n for any risk function under which the expected work production of the group schedule within the charged-initiation model is a *unimodal* function of n. (It is quite natural to assume that this expectation is nondecreasing under the free-initiation model, but not with the charged-initiation model where the per-chunk overhead damps the work production.) We can then use binary search (on the number of chunks per slice) to seek the optimum value of n; the search can be safely performed in the interval $[1..X/\varepsilon]$.

5.2.4 Group Schedules: A Sampler

Our group schedules strive to minimize the impact of work-killing interruptions—and, thereby, to maximize expected work completion—by having every computer attempt to compute every chunk. Of course, there are many ways to achieve this coverage, and the form of the risk function will make some ways more advantageous than others. As an extreme example, when $p = 2$, it is *always* advantageous to have the remote computers

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

(a) Cyclic: $\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} = 34104$

1	2	3	4	5
10	9	8	7	6
15	14	13	12	11
20	19	18	17	16

(b) Reverse: $\mathsf{K}^{(\Sigma_{\mathsf{reverse}})} = 24396$

1	2	3	4	5
6	7	8	9	10
15	14	13	12	11
20	19	18	17	16

(c) Mirror: $\mathsf{K}^{(\Sigma_{\mathsf{mirror}})} = 27284$

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16

(d) Snake: $\mathsf{K}^{(\Sigma_{\mathsf{snake}})} = 25784$

1	2	3	4	5
14	12	10	8	6
15	13	11	9	7
16	17	18	19	20

(e) Fat-snake: $\mathsf{K}^{(\Sigma_{\mathsf{fat\text{-}snake}})} = 24276$

1	2	3	4	5
10	9	8	7	6
15	14	13	12	11
20	19	18	16	17

(f) Greedy: $\mathsf{K}^{(\Sigma_{\mathsf{greedy}})} = 24390$

1	2	3	4	5
13	10	6	9	7
18	15	14	11	8
20	16	19	12	17

(g) Optimal: $\mathsf{K}^{(\Sigma_{\mathrm{Optimal}})} = 23780$

TABLE 3
Comparing group schedules for $n = 20$ and $g = 4$. Here the most efficient group schedules are $\Sigma_{\mathsf{fat\text{-}snake}}$, Σ_{greedy}, and $\Sigma_{\mathsf{reverse}}$ (in this order). The lower bound, $\mathsf{K}_{\min} = 23780$, is reached on this example.

process replicated work "in opposite orders" (Theorem 5). We now specify and compare the performance of five group schedules whose chunk-scheduling regimens seem to be a good match for the way the linear risk function "predicts" interruptions. We specify each schedule Σ via its group execution chart $\widehat{C}^{(\Sigma)}$—see Table 3—and we represent the performance of each schedule Σ via its performance constant $\mathsf{K}^{(\Sigma)}$. The beneficent structures of these schedules is evidenced by our ability to present explicit symbolic expressions for their K constants.

Cyclic scheduling (See Table 3(a)). Under this simplest scheduling regimen, whose schedules we denote by Σ_{cyclic}, groups are executed sequentially in a round-robin fashion, so the chunks of group j are executed at steps j, $j + n/g$, $j + 2n/g$, and so on. One verifies that

$$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} \;=\; \sum_{j=1}^{n/g} \prod_{k=0}^{g-1} (j + kn/g)\,.$$

The weakness of Σ_{cyclic} is that chunks in low-index groups have a higher probability of being completed successfully than do chunks in high-index groups—because chunks remain in the same relative order throughout the computation. The other schedules we consider aim to compensate for this imbalance.

Reverse scheduling (See Table 3(b)). Under this regimen, whose schedules we denote by $\Sigma_{\mathsf{reverse}}$, the chunks in each group are executed once in the initially specified order and then in the *reverse* order $n/g - 1$ times. Thus, the chunks in group j are executed at step j and thereafter, at steps $2n/g-j+1$, $3n/g-j+1$, $4n/g-j+1$, and so on. $\Sigma_{\mathsf{reverse}}$ is motivated by an attempt to compensate for the imbalance in chunks' likelihoods of being completed created by their initial order of processing. ($\Sigma_{\mathsf{reverse}}$ is the schedule specified in Table 2.) One verifies that

$$\mathsf{K}^{(\Sigma_{\mathsf{reverse}})} \;=\; \sum_{j=1}^{n/g} j \times \prod_{k=1}^{g-1} ((k+1)n/g - j + 1)\,.$$

Mirror scheduling (See Table 3(c)). This regimen, whose schedules we denote by Σ_{mirror}, represents a compromise between the cyclic and reverse regimens. Σ_{mirror} compensates for the imbalance in the likelihood of work's being completed only during the second half of the computation. Specifically, Σ_{mirror} mimics Σ_{cyclic} for the first $g/2$ phases of processing a group, and it mimics $\Sigma_{\mathsf{reverse}}$ for the remaining phases. One verifies that

$$\mathsf{K}^{(\Sigma_{\mathsf{mirror}})} \;=\; \sum_{j=1}^{n/g} \prod_{k=0}^{\frac{1}{2}g-1} (j + kn/g)\,((p-k)n/g - j + 1)\,.$$

Snake-like scheduling (See Table 3(d)). This regimen, whose schedules we denote by Σ_{snake}, compensates for the imbalance of the cyclic schedule by mimicking Σ_{cyclic} at every odd-numbered step and $\Sigma_{\mathsf{reverse}}$ at every even-numbered step, thereby lending a snake-like structure to its execution chart, $\widehat{C}^{(\Sigma_{\mathsf{snake}})}$. One verifies easily that

$$\mathsf{K}^{(\Sigma_{\mathsf{snake}})} \;=\; \sum_{j=1}^{n/g} \prod_{k=0}^{\frac{1}{2}g-1} (j + 2kn/g)\,(2(k+1)n/g - j + 1)\,.$$

Fat-snake-like scheduling (See Table 3(e)). This regimen, whose schedules we denote by $\Sigma_{\mathsf{fat\text{-}snake}}$, qualitatively adopts the same strategy as Σ_{snake}, but it slows down the reverse phase of the latter regimen. Consider, for illustration, three consecutive rows of $\widehat{C}^{(\Sigma_{\mathsf{fat\text{-}snake}})}$. The first row is identical in shape to the first row of $\widehat{C}^{(\Sigma_{\mathsf{cyclic}})}$. The reverse phase of Fat-snake distributes the elements of the two remaining rows in the reverse order, two elements at a time. The motivating intuition is that the slower reversal would further compensate for the imbalance in Σ_{cyclic}. One verifies that

$$\mathsf{K}^{(\Sigma_{\mathsf{fat\text{-}snake}})} \;=\; \sum_{j=0}^{n/g-1} \prod_{k=0}^{\frac{1}{3}g-1} (1+j+3k\frac{n}{g})(3(k+1)\frac{n}{g}-2j-1)(3(k+1)\frac{n}{g}-2j).$$

We derive performance bounds for these five scheduling regimens.

	Relative				Absolute				Success rate
	min	max	avg.	stdv.	min	max	avg.	stdv.	
Cyclic	1.1	3.786	2.143	0.664	1.1	3.786	2.239	0.592	00.00%
Reverse	1	1.295	1.055	0.065	1	1.295	1.117	0.061	12.42%
Mirror	1	2.468	1.504	0.393	1	2.468	1.575	0.338	12.37%
Snake	1	1.199	1.127	0.059	1	1.291	1.193	0.059	12.34%
Fat-snake	1	1.442	1.123	0.115	1	1.530	1.192	0.143	17.07%
Greedy	1	1.055	1.005	0.015	1	1.224	1.067	0.074	83.01%
Best-of	1	1	1	0	1	1.224	1.061	0.069	100.00%

TABLE 4
Statistics on the K value of all heuristics for $2 \leq g \leq 100$ and $2g \leq n \leq 1000$ (minimum, maximum, average value and standard deviation over the 4032 instances).

Proposition 2: The values of $\mathsf{K}^{(\Sigma)}$ for our five scheduling regimens — presented as the ratio $\mathsf{K}^{(\Sigma)} \div (g!\,(n/g)^{g+1})$ — satisfy the following bounds:

Regimen	Lower bound	Upper bound
$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})}$	$1/n$	1
$\mathsf{K}^{(\Sigma_{\mathsf{reverse}})}$	$1/(2g)$	$\frac{1}{2}(n+g)$
$\mathsf{K}^{(\Sigma_{\mathsf{mirror}})}$	$1/n$	1
$\mathsf{K}^{(\Sigma_{\mathsf{snake}})}$	g/n^2	1
$\mathsf{K}^{(\Sigma_{\mathsf{fat\text{-}snake}})}$	$1/((g-1)n)$	g

Proof: The calculations are straightforward. For Σ_{cyclic}, we have

$$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} = \sum_{j=1}^{\frac{n}{g}} \prod_{k=0}^{g-1} \left(j + k\frac{n}{g} \right).$$

We derive the lower bound by replacing index j by 0 in the summation (except in the term $k=0$ where we replace j by 1):

$$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} \geq \frac{n}{g} \left(\prod_{k=1}^{g-1} k\frac{n}{g} \right) = (g-1)! \left(\frac{n}{g}\right)^{g} = \frac{1}{n} g! \left(\frac{n}{g}\right)^{g+1}.$$

Similarly, we let $j = \frac{n}{g}$ in each term of the summation to get the upper bound. We proceed in a similar way for the other three variants.

We explicit the computations for $\Sigma_{\mathsf{fat\text{-}snake}}$, as they are a bit less obvious. For the lower bound we have:

$$\begin{aligned}
&\mathsf{K}^{(\Sigma_{\mathsf{fat\text{-}snake}})} \\
&= \sum_{j=0}^{\frac{n}{g}-1} \prod_{k=0}^{\frac{g}{3}-1} \left(1+j+3k\frac{n}{g}\right)\left(3(k+1)\frac{n}{g}-2j-1\right)\left(3(k+1)\frac{n}{g}-2j\right) \\
&\geq \sum_{j=0}^{\frac{n}{g}-1} \prod_{k=0}^{\frac{g}{3}-1} \left(1+3k\frac{n}{g}\right)\left(3(k+1)\frac{n}{g}-2\frac{n}{g}+1\right)\left(3(k+1)\frac{n}{g}-2\frac{n}{g}+2\right) \\
&\geq \tfrac{n}{g} \textstyle\prod_{k=0}^{\frac{g}{3}-1} \left(1+3k\frac{n}{g}\right)\left((3k+1)\frac{n}{g}\right)\left((3k+1)\frac{n}{g}\right) \\
&\geq \left(\tfrac{n}{g}\right)^3 \textstyle\prod_{k=1}^{\frac{g}{3}-1} \left(3k\frac{n}{g}\right)\left((3k+1)\frac{n}{g}\right)\left((3k+1)\frac{n}{g}\right) \\
&\geq \left(\tfrac{n}{g}\right)^3 \textstyle\prod_{k=1}^{\frac{g}{3}-1} \left((3k-1)\frac{n}{g}\right)\left(3k\frac{n}{g}\right)\left((3k+1)\frac{n}{g}\right) \\
&= \left(\tfrac{n}{g}\right)^g (g-2)!
\end{aligned}$$

For the upper bound we derive:

$$\begin{aligned}
&\mathsf{K}^{(\Sigma_{\mathsf{fat\text{-}snake}})} \\
&= \sum_{j=0}^{\frac{n}{g}-1} \prod_{k=0}^{\frac{g}{3}-1} \left(1+j+3k\frac{n}{g}\right)\left(3(k+1)\frac{n}{g}-2j-1\right)\left(3(k+1)\frac{n}{g}-2j\right) \\
&\leq \sum_{j=0}^{\frac{n}{g}-1} \prod_{k=0}^{\frac{g}{3}-1} \left(\frac{n}{g}+3k\frac{n}{g}\right)\left(3(k+1)\frac{n}{g}\right)\left(3(k+1)\frac{n}{g}\right) \\
&\leq \frac{n}{g} \prod_{k=0}^{\frac{g}{3}-1} \left((3k+1)\frac{n}{g}\right)\left((3k+3)\frac{n}{g}\right)\left((3k+3)\frac{n}{g}\right) \\
&\leq \left(\frac{n}{g}\right)^{g+1} \left(\prod_{k=0}^{\frac{g}{3}-2} (3k+2)(3k+3)(3k+4) \right) (g-2)g^2 \\
&= \left(\frac{n}{g}\right)^{g+1} (g-2)!(g-2)g^2 \leq \left(\frac{n}{g}\right)^{g+1} (g)!\,g
\end{aligned}$$

□

Proposition 2 suggests that Σ_{snake} may be the most efficient of our five group-scheduling algorithms, especially when we checkpoint often, i.e., when n is large. While still focusing on mathematical analyses of our schedules, though, we can use Stirling's formula to derive the following more evocative bounds on $\mathsf{K}^{(\Sigma_{\mathsf{snake}})}$:

$$\mathsf{K}_{\min} \;\leq\; \mathsf{K}^{(\Sigma_{\mathsf{snake}})} \;\leq\; \mathsf{K}_{\mathsf{upper}},$$

where

$$\mathsf{K}_{\min} \approx \frac{e}{g}\left(\frac{n}{g}\right)^{g+1} \text{ and } \mathsf{K}_{\mathsf{upper}} = g!\left(\frac{n}{g}\right)^{g+1} \approx \frac{e\sqrt{2\pi}}{\sqrt{g}}\left(\frac{n}{e}\right)^{g+1}.$$

We conclude this subsection by adding a last group schedule to our list, as a reference point for our experiments. The *"greedy"* scheduling regimen, whose schedules we denote by Σ_{greedy}, strives to iteratively balance the probability of successful completion for each group of chunks. We have not been able to derive an analytical asymptotic estimate of Σ_{greedy}'s expected work production, so we content ourselves with a numerical estimate. Greedy scheduling (See Table 3(f)). This regimen, whose schedules we denote by Σ_{greedy}, iteratively assigns a step to each group of chunks so as to balance the current success probabilities as much as possible. At each step, Σ_{greedy} constructs one new row of its execution chart

$\widehat{C}^{(\text{greedy})}$. Thus, after k steps, the probability that a chunk in group j will be interrupted is proportional to the product $\prod_{i=1}^{k} \widehat{C}_{ij}^{(\text{greedy})}$ of the entries in column j of the chart. The idea is to sort current column products and assign the smallest time-step to the largest product, and so on.

Table 3 provides a scenario in which Σ_{greedy} outperforms most of the other heuristics. None of our group schedules completes an optimal amount of work in this scenario, which shows that one pays a price for the heuristics' simple structure.

5.3 Evaluating the Heuristics Numerically

We ran all six of our scheduling heuristics on problems having all parameter values in the following ranges: $g \in [2, 100]$, $n \in [2g, 1000]$, and g divides n, altogether 4032 instances. Table 4 summarizes the results of this evaluation via two series of statistics. In the *Relative* series, we form the ratio of the performance constant K of a given heuristic on a given problem instance and (i.e., divided by) the lowest value of K that we found for that instance, among all the tested heuristics. For the *Absolute* series, we form the analogous ratio with $\mathsf{K}_{\min}$ in the denominator. For perspective, the table also reports the "performance" of the (unrealistic) *best-of* heuristic that, on each problem instance, runs the six other algorithms and picks the best answer.

Table 4 shows that Σ_{greedy} is clearly the best heuristic: it finds the best schedule for 83% of the problem instances, and its schedules are never more than 6% worse than the best one found. More importantly, Σ_{greedy}'s performance is never more than 23% larger than the lower bound $\mathsf{K}_{\min}$ and, on average, it is less than 7% larger than this bound. In fact, only $\Sigma_{\text{fat-snake}}$ happens sometimes to find better solutions than Σ_{greedy}—but only by marginal amounts, as one can see by comparing the absolute performance of Σ_{greedy} and *best-of*.

6 Experiments Based on the Linear Risk Model

We have assessed the performance of our group heuristics by testing them on simulated computing platforms that are subject to unrecoverable interruptions (see Appendix B). Since the relative performance of the group heuristics were consistent with the theoretical predictions of Table 4, we carry forward only the champion heuristic, Σ_{greedy}, to the experiments that follow. Whereas the heuristics of Section 5 (including Σ_{greedy}) were tailored for the linear risk model, the heuristics that we study henceforth as competitors for Σ_{greedy} are suggested less formally by the way they implement the strategy of work replication. The source codes and raw data for all heuristics can be found at http://graal.ens-lyon.fr/~fvivien/Publications/Data/Interruptions. The appendices present the graphs for all the experiments.

6.1 The Experimental Plan

We performed our experiments on randomly generated platforms containing p computers. In all experiments, we set $\kappa = 1$ and chose the times for interruptions randomly between 0 and 1, following a uniform distribution. We varied the size of the overall workload, $W_{(\text{ttl})}$, between 1 and p. The case $W_{(\text{ttl})} = 1$ represents the scenario in which all computers can potentially do all the work before being interrupted. The case $W_{(\text{ttl})} = p$ represents the scenario when we can do no better than give one unit-size slice to each computer, which then computes until it is interrupted; there is no replication in this case.

The key parameters in our experiments are: the number of computers p; the total amount of work $W_{(\text{ttl})}$; the number of chunks per unit of work n; and the start-up cost ε. In the first four series of experiments, three of these parameters are fixed while the fourth is varied. When fixed, these parameters take the following values:

- $p \in \{5, 10, 25, 50, 100\}$;
- $W_{(\text{ttl})} = 0.3p$ or $0.7p$;
- $n \in \{47, 97, 147, 197\}$;
- $\varepsilon \in \{0.1000, 0.0100, 0.0010, 0.0001\}$.

We have the parameters range over large sets of values in order to assess the heuristics in very different configurations, even very unfavorable ones.

Our experiments compared the performance of the following heuristics:

The group-greedy heuristic Σ_{greedy}. We have already seen Σ_{greedy} in Section 5.2.2. Since the number of chunks n may not be a multiple of g, the last group of computers may not have a full set of g chunks to process. This heuristic works *as though* the last group contained g chunks; this has the potential impact of inserting idle time-slots in a schedule. (A computer that still has work will never be kept idle.) The values for n were explicitly picked *not* to favor the group-heuristics: they almost certainly ensured that the last group of computers never had a full set of g chunks to process.

The brute-force replication heuristic Σ_{brute} replicates the entire workload onto all computers. Each computer executes work in the order of receipt, starting from the first chunk, until it is interrupted.

The no-replication heuristic $\Sigma_{\text{no-rep}}$ distributes the work in a round-robin fashion, with no replication. Thus, each computer is allocated $W_{(\text{ttl})}/p$ units of work.

The cyclic-replication heuristic $\Sigma_{\text{cyclic-rep}}$ distributes the work in a round-robin fashion, as does $\Sigma_{\text{no-rep}}$, but it keeps distributing chunks, starting from chunk 1 again, until each computer has a total (local) workload of 1. Note that when the number of chunks is a multiple of p, this heuristic is identical to $\Sigma_{\text{no-rep}}$, because the chunks assigned to a computer during the replication phase were already assigned to it previously.

The random-replication heuristic $\Sigma_{\text{random-rep}}$ distributes a total workload of 1 to each computer, but it chooses the chunks and their order randomly, while ensuring that all chunks deployed on any given computer are distinct.

Note, however, that the same chunk can be assigned to several computers.

The omniscient heuristic $\Sigma_{\text{omniscient}}$ is an idealized, unrealizable "heuristic" that is included only as a reference point. The "heuristic" knows (in advance) exactly when each computer will be interrupted. It deploys on each computer a single chunk of length $\ell + \varepsilon$ (recall that ε is the start-up cost); ℓ is computed so that the computer completes its work immediately before it is interrupted. Thus, this heuristic, knowing all interruption times, completes the maximum possible amount of work.

We do not report the absolute amount of work completed by the heuristics: This quantity would be impossible to interpret because the amount of work deployed and the amount of work that can be completed before computers are interrupted vary vastly within the experiment. We therefore consider, for each instance, the *ratio of the work completed by a given heuristic and the work completed by* $\Sigma_{\text{omniscient}}$. Since $\Sigma_{\text{omniscient}}$ always achieves a performance ratio of 1, we do not display it on figures. (By convention, in the marginal cases where $\Sigma_{\text{omniscient}}$ does not complete any work—because all computers are interrupted by time ε—the performance of all heuristics is set to 1.)

6.2 Results from Idealized Experiments

For each considered set of parameters, we randomly generated 1000 different sets of interruption times. In Experiments E1–E4, we fixed all parameters but one; Experiment E5 studies the effectiveness of the procedure from Section 5.2.3 for choosing the sizes of the chunks we deploy; Experiment E6 studies the impact of work replication. Figure 4 presents the result of Experiments E1-E4 for a sample set of parameters. (Graphs showing the impact of the various parameters are available appendices A and B.)

Experiment E1: varying workload size. When $W_{(\text{ttl})} = 1$, the opportunities for replicating work are maximum. As one would expect in this case, $\Sigma_{\text{random-rep}}$ often dominates $\Sigma_{\text{no-rep}}$. Replication is therefore worth considering—but it must be implemented in a meaningful way: Σ_{brute} almost always achieves very poor performance. Another obvious conclusion is that when there is very little room for replication, i.e., when $W_{(\text{ttl})}$ is close to p, $\Sigma_{\text{no-rep}}$, $\Sigma_{\text{cyclic-rep}}$, and Σ_{greedy} achieve quite similar performance. In all cases, $\Sigma_{\text{cyclic-rep}}$ achieves better performance than $\Sigma_{\text{no-rep}}$. This is significant when $W_{(\text{ttl})}$ is small relative to pX. On every instance, Σ_{greedy} exhibits the best performance.

Experiment E2: varying number of computers. The performance of our heuristics is generally not impacted when the number of computers p grows while the overall work-to-computer ratio $W_{(\text{ttl})}/p$ is kept constant. (The exception is Σ_{brute} whose performance drops dramatically.) As the ratio $W_{(\text{ttl})}/p$ increases, there is less opportunity for replication, so efficient use of resources becomes more complicated; this leads to overall worse performance by all of the heuristics.

Experiment E3: varying number of chunks. When the start-up cost ε is very small, one is always better off deploying work in a larger number of chunks (i.e., more checkpointing), because having small chunks reduces the loss of work in progress when a computer is interrupted. However, for larger values of ε, one must be more cautious in choosing chunk sizes, because the "penalty" for each additional chunk/checkpoint then negatively impacts the expected work production; when ε is large, this impact is dramatic. It is not clear that the loss of expected work would be significant in an intermediate case such as $\varepsilon = 0.001$. But even in this case, the performance starts to decrease as the number of chunks increases. Of course, one must exercise special care in choosing the exact number of chunks when scheduling using $\Sigma_{\text{cyclic-rep}}$, for that heuristic's performance fluctuates depending on whether the number of computers is relatively prime to the number of chunks. As a positive sidenote from this experiment, the general shapes of the performance curves corroborate the unimodal assumption proposed at the end of Section 5.2.3.

Because the studied parameter is not *the overall number of chunks* but, rather, *the number of chunks per unit of work*, the number of computers has no significant impact on performance (except, obviously, for $\Sigma_{\text{cyclic-rep}}$).

Experiment E4: varying the start-up cost ε. As one considers successively larger values of ε, one observes a dramatic drop in performance. Indeed, when ε is large, e.g., when $\varepsilon \geq 0.05$ (roughly), very few chunks can be executed on a computer before it is interrupted. In these configurations, performance depends mainly on the size of chunks relative to the interruption times in the instance. There is no way to design good heuristics on average (compared to $\Sigma_{\text{omniscient}}$) and all heuristics have poor average performance. This effect gets even worse with larger numbers of chunks per unit of work. Indeed, as ε approaches 1, the proportion of cases where even $\Sigma_{\text{omniscient}}$ does not complete any work increases. In these pathological cases, all of our heuristics have performance ratios of 1.

(Due to the poor performance of $\Sigma_{\text{random-rep}}$, we do not consider this heuristic in the following experiments.)

Experiment E5: automatic inference of chunk size. This experiment replicates experiment E1 except that, for each instance and each heuristic, the size of chunks is no longer given but is, rather, automatically inferred using the procedure of Section 5.2.3. In Fig. 5, for each heuristic, we plot the average performance when considering only the x% best instances for that heuristic. The performance for 100% is thus the average performance over all 760,000 instances. One observes Σ_{greedy} achieving at least 85.2% of the optimal work production of $\Sigma_{\text{omniscient}}$, with $\Sigma_{\text{no-rep}}$ close behind with 79.6% of the optimal. Thus, on average, Σ_{greedy} is 27.4% closer to optimal than is $\Sigma_{\text{no-rep}}$. Furthermore, in more than 21% of the instances, Σ_{greedy} is almost optimal, achieving more than 99.5% of the optimal work production. $\Sigma_{\text{cyclic-rep}}$'s work production is quite close to Σ_{greedy}'s.

Experiment E6: varying ratio of potential replication. This experiment is dedicated to assessing the impact of the ratio of potential replication $pX/W_{(\mathrm{ttl})}$. We fixed the overall workload to $W_{(\mathrm{ttl})} = 10$ units and allowed p to assume all integral values between 10 and 100 (with the earlier mentioned four choices for the value of ε). We considered 1000 random instances of each set of parameters. The results are presented in Fig. 6. $\Sigma_{\text{cyclic-rep}}$ and Σ_{greedy} always have better performance than $\Sigma_{\text{no-rep}}$, and the difference is very significant as soon as the ratio of potential replication exceeds 2. Overall, Σ_{greedy} has better and more regular performance than $\Sigma_{\text{cyclic-rep}}$. In contrast with Σ_{greedy}, $\Sigma_{\text{cyclic-rep}}$ takes almost no advantage of the possibility of replication when the potential for replication is smaller than 2.

Summarizing the idealized experiments. The experiments show that careful use of work replication can significantly improve the performance of heuristics. The greedy heuristic Σ_{greedy} always delivers good performance, it is never outperformed by any other heuristic—on average it delivers the best performance on every configuration—and in favorable cases, it performs significantly better than any other heuristic.

7 Experiments Based on Actual Traces

Most of the results in this paper focus on the linear risk model. Three notable exceptions that are relevant to the current study are the following results that relate to scheduling for arbitrary nondecreasing risk functions: (1) Theorem 5, which supplies guidelines for crafting optimal schedules for two remote computers under the free-initiation model; (2) Theorem 9, which exposes a close relationship between the expected work production under the free- and the charged-initiation models; (3) Theorem 6 which describes an asymptotically optimal schedule for two remote computers under the free-initiation model. Inspired by the cited theoretical results and by the development in Section 5.2.2, we devote Section 7.1 to developing group-scheduling heuristics that are intended for use with arbitrary risk functions. We then evaluate these heuristics using actual traces, in Section 7.2.

7.1 Heuristics for Arbitrary Interruption Risk

Based on our work to this point, it is natural to restrict attention to schedules that deploy equal-size chunks when scheduling for the general setting of p remote computers under arbitrary nondecreasing risk functions. First, we know that such equal-size-chunk schedules are asymptotically optimal when scheduling one or two remote computers; Theorem 3 and Theorem 6. Second, the group-scheduling heuristics of Section 5.2 deploy work in equal-size chunks, and this class of schedules are structurally quite attractive as we contemplate how to deploy work to arbitrary numbers of computers.

Elaborating on the second point, we note that, with the sole exception of the greedy scheduling heuristic, the underlying risk function does not play any role in the definition of any of our group-scheduling heuristics. The underlying risk function *does* have an impact on the choice of the optimal number of chunks (see Section 5.2.3) but that is the function's only impact. Therefore, adapting any of our group-scheduling heuristics (other than the greedy heuristic) to another risk function requires only changing the number of chunks that the heuristic works with.

The preceding is both good and bad news: it is good news since adapting almost all heuristics is easy; it is bad news because the sole exception is the heuristic that dominated in all of our tests. Therefore, it is worthwhile working a bit to adapt the greedy scheduling heuristic for arbitrary interruption risk. In fact, this is not so hard: after k steps, the probability that a chunk in group j will be interrupted is proportional to the product $\prod_{i=1}^{k} Pr(\widehat{C}_{ij}^{(\text{greedy})} sl/n)$. We can, therefore, adapt the greedy scheduling heuristic by using this general expression for the probability, in place of the expression specialized for the linear risk model, i.e., $\prod_{i=1}^{k} \widehat{C}_{ij}^{(\text{greedy})}$.

We are now ready to assess the quality of our heuristics, using actual activity traces.

7.2 Trace-based Experiments

7.2.1 Traces and Methodology

We evaluate our scheduling heuristics using eight traces that recorded, for each computer in an assemblage, the lengths of the different time interval during which the computer was available. These traces are: 0) the *SDSC trace* [26, p. 33] with 5678 availability durations; 1) the *UCB trace* [27] with 19276 availability durations; 2) the *XtremWeb trace* [26, p. 33] with 8756 availability durations; 3) the *Cetus trace* [28] with 1898 availability durations; 4) the *LONG trace* [28] with 10958 availability durations; 5) the *Princeton trace* [28] with 79 availability durations; 6) the *Condor trace* [29] with 1125 availability durations; and 7) the *CSIL trace* [29] with 927 availability durations.

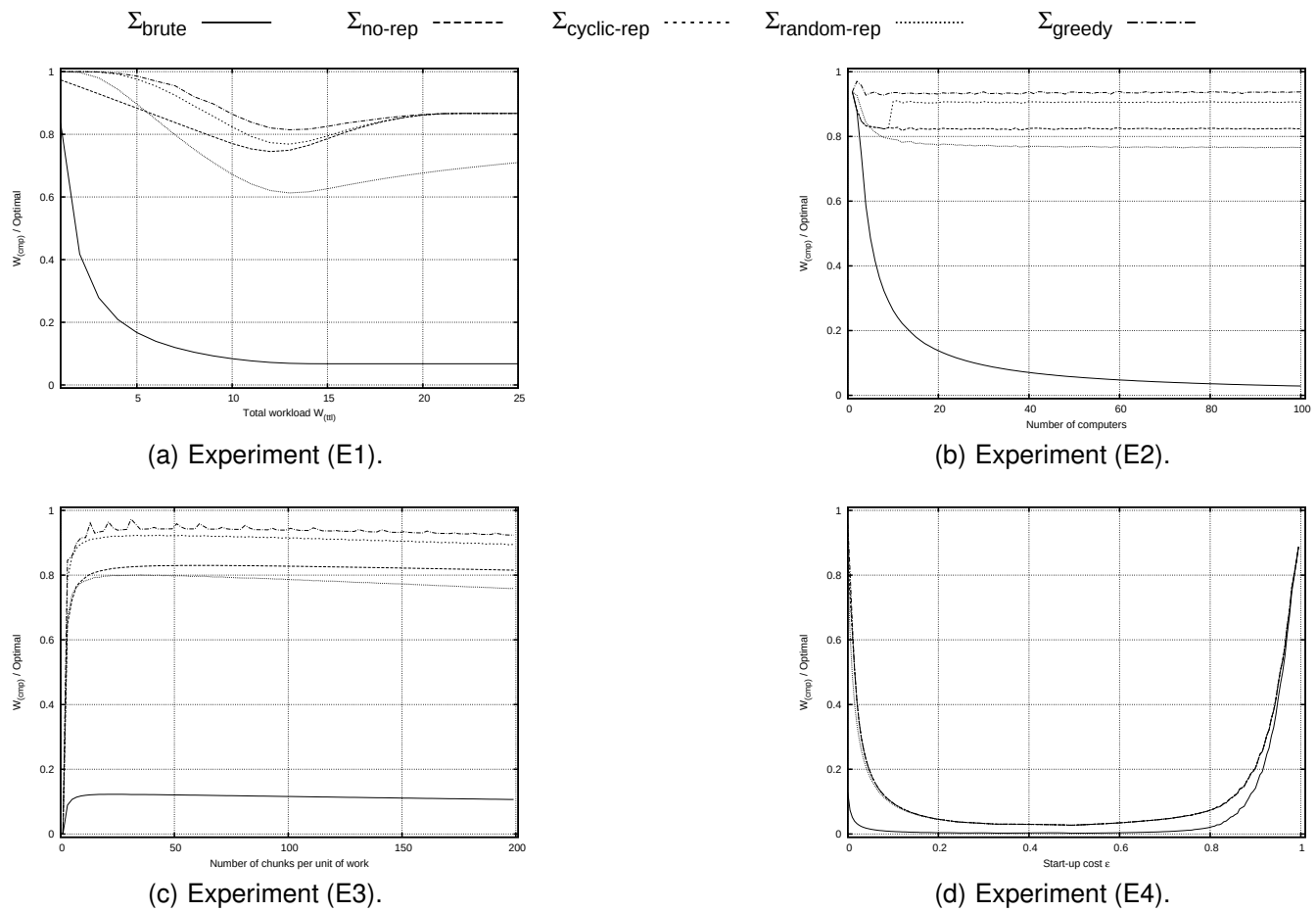
We normalized these traces so that the longest availability interval for each trace is exactly 1. (This allows us to compare and average statistics over different traces). Then, from each trace, *trace*, we built a risk function, $Pr_{trace}(t)$, that specifies the probability of a computer's being interrupted by time t:

$$Pr_{trace}(t) = \frac{\text{Number of intervals in } \textit{trace} \text{ shorter than } t}{\text{Number of intervals in } \textit{trace}}.$$

We generated interruption instances by uniformly and randomly picking availability interval lengths in the studied trace. Therefore, we assumed implicitly that when making a scheduling decision, we only considered computers that just became available.

7.2.2 Simulation Results

Parameter settings. We ran the heuristics with parameter λ (see Section 5.1) set to 1.00, parameters p and ε set as



(a) Experiment (E1). (b) Experiment (E2).

(c) Experiment (E3). (d) Experiment (E4).

Fig. 4. Experiments (E1) through (E4) with 25 computers, 147 chunks, $\varepsilon = 0.0010$, and $W_{(\mathrm{ttl})} = 0.3p$.

in Section 6.1, and parameter $W_{(\mathrm{ttl})}$ taking all integral values in the range $[1..p]$. For each set of parameters and each trace we generated 1,000 interruption scenarios.

Results. The aggregated simulation results are presented in Fig. 7 and Table 5. (Graphs showing the impact of the various parameters are available in appendix C.) Overall, and under each studied scenario, Σ_{greedy} achieves far better results than $\Sigma_{\mathsf{cyclic\text{-}rep}}$ and $\Sigma_{\mathsf{no\text{-}rep}}$. The difference between Σ_{greedy} and the other heuristics becomes more and more important as the number of computers increases or as the size of the start-up cost decreases: the more freedom for work replication, the more obvious the advantage of Σ_{greedy}.

Fig. 8 presents aggregated result for each of the eight traces, and Fig. 9 presents the risk function of each of the eight traces. Trace 5 is the most similar and Trace 1 the least similar to the linear risk model. For both traces, and for all the intermediate cases, Σ_{greedy} achieves the best overall performance with a significant margin. This is the case when a lot of work can be successfully processed (Traces 0, 3, 5, and 7, for which the performance of Σ_{brute} is below 20%). This is also the case when very little work can be performed before all processors are interrupted (Traces 1 and 2).

The performance of $\Sigma_{\mathsf{cyclic\text{-}rep}}$ is close to that of $\Sigma_{\mathsf{no\text{-}rep}}$. On average, Σ_{greedy} closes 37% of the gap between $\Sigma_{\mathsf{no\text{-}rep}}$ and the (omniscient) optimal heuristic.

The comparison of $\Sigma_{\mathsf{omniscient}}$ and Σ_{greedy} shows that the latter has very good absolute performance. This proves the efficiency of static heuristics in this context.

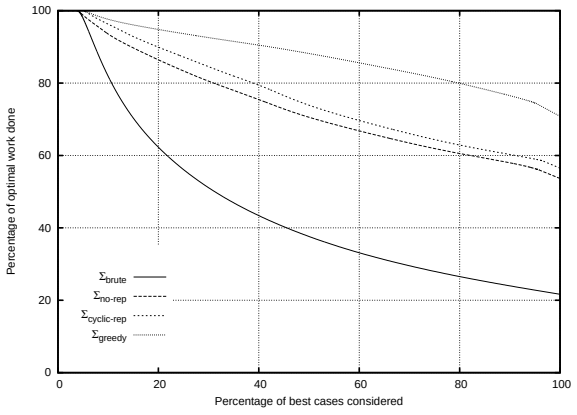


Fig. 7. Performance of the heuristics with risk functions defined by computer-availability traces. For each of the four heuristics, the curve $y = f(x)$ shows the average performance y of the heuristics when only considering the $x\%$ cases most favorable to that heuristic.

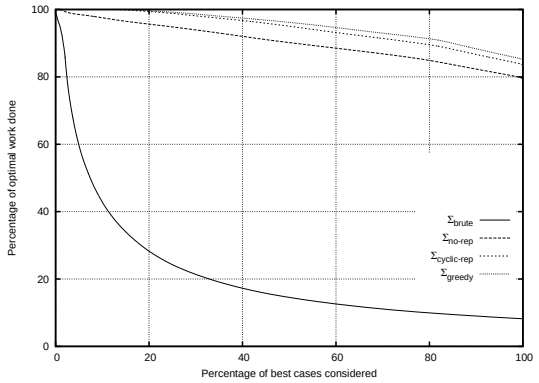


Fig. 5. Experiment E5: performance with automatic inference of chunk sizes.

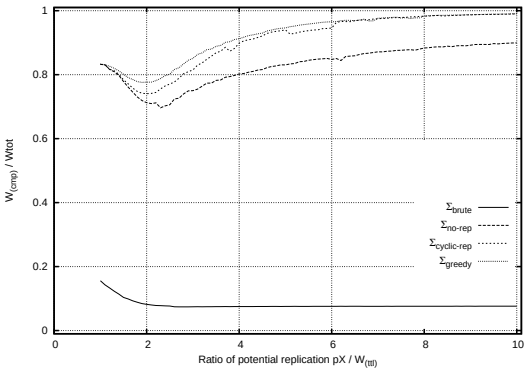


Fig. 6. Experiment E6: impact of the ratio of potential replication, $pX/W_{(\mathrm{ttl})}$.

Heuristic	average	stdv
Σ_{brute}	21.7	24.2
$\Sigma_{no\text{-}rep}$	53.6	22.1
$\Sigma_{cyclic\text{-}rep}$	56.5	22.2
Σ_{greedy}	70.8	23.2

TABLE 5
Aggregate performance over all 6,080,000 instances for risk functions defined by computer-availability traces.

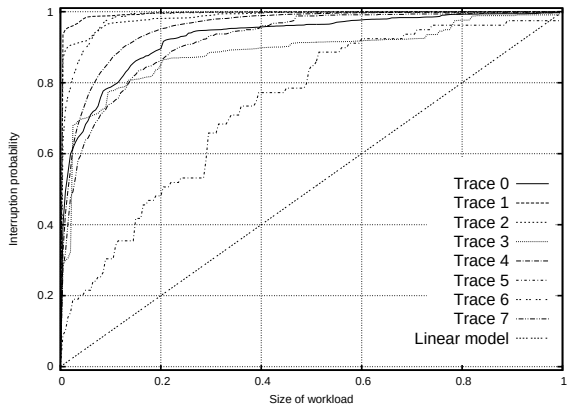


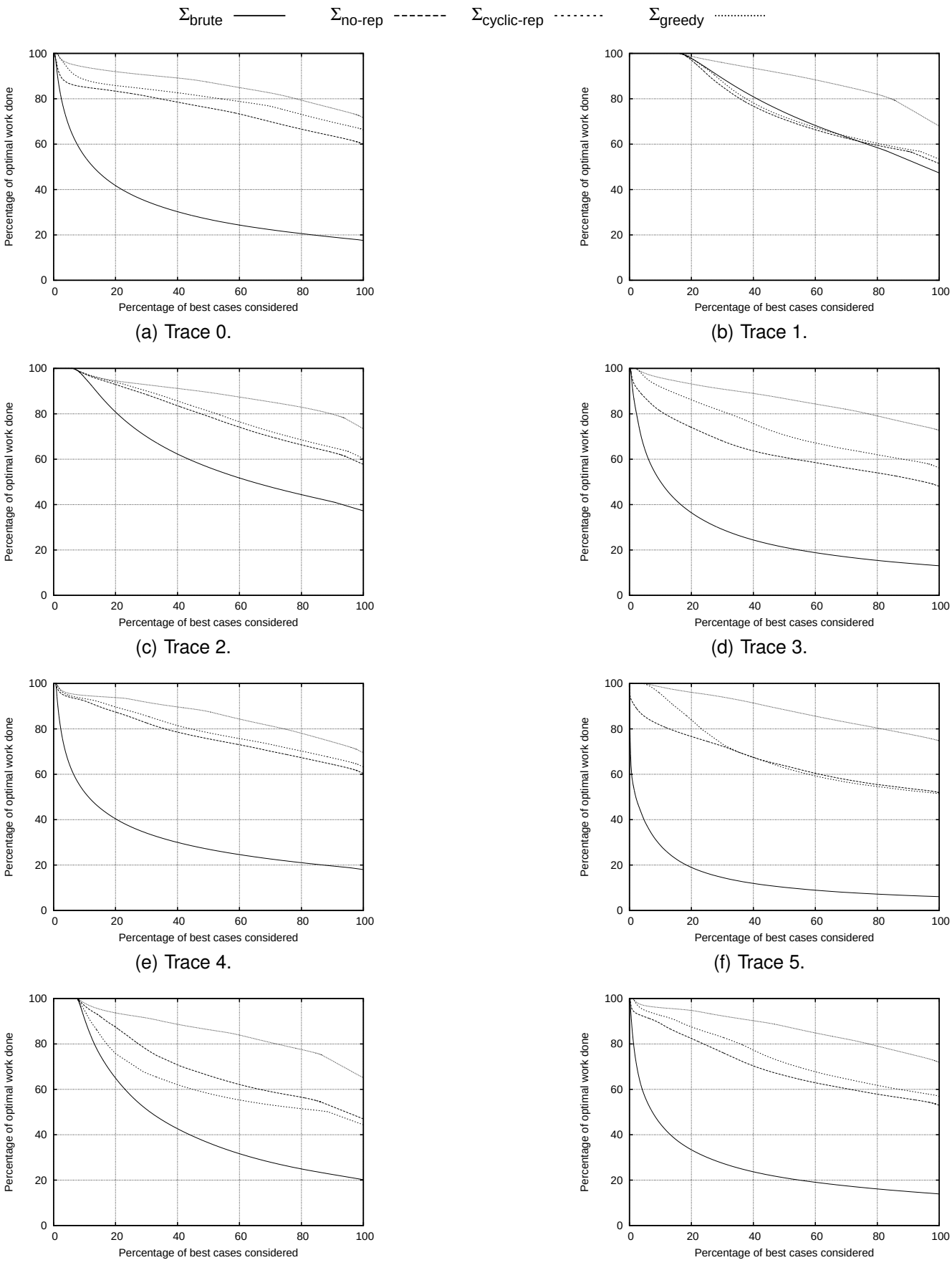
Fig. 9. Interruption probability for each of the eight traces.

8 Conclusion

We have presented a model for studying the problem of scheduling large divisible workloads on identical remote computers that are vulnerable (with the same risk function) to unrecoverable interruptions. Our goal has been to find schedules for allocating work to the computers and for scheduling the checkpointing of that work, in a manner that maximizes the expected amount of work completed by the remote computers. Most of the results we report assume that the risk of a remote computer being interrupted increases *linearly* with the amount of time that the computer has been available to us; a few results provide scheduling guidelines for more general risk functions.

We have completely solved this scheduling problem for the case of $p = 1$ remote computer (see [1] and Section 3), by deriving schedules whose expected work completion is exactly maximum, both for the free-initiation model, wherein checkpointing incurs no overhead, and the charged-initiation model, wherein checkpointing does incur an overhead. Our major focus is the case of $p = 2$ remote computers (Section 4). We provide schedules for this case whose expected work completion is either exactly or asymptotically optimal when the size of the workload grows without bound; we also provide guidelines for deriving exactly optimal schedules. The complexity of the development in Section 4 suggests that the general case of p remote computers will be prohibitively difficult, even for deriving asymptotically optimal schedules. Therefore, we settle in this general case for deriving a number of well-structured heuristics, whose quality can be assessed via explicit expressions for their expected work outputs (Section 5). Numerical evaluations suggest that one of our six group heuristics is the winner in terms of performance. An extensive suite of simulation experiments suggests that the "winner" in the competition of Section 5 provides schedules with good expected work output, and that it dominates the reference heuristics (Section 6). Finally, building on the insight gained studying the linear risk model, we turned our attention to general risk models (Section 7). We adapted to general risk functions our p-computer heuristics. Extensive simulations using actual traces of computer availabilities suggest that the winner of Sections 5 and 6 also dominate reference heuristics in the presence of general risk functions, and is a very efficient heuristic. Furthermore, these simulations show that static heuristics have overall very good absolute performance.

Much remains to be done regarding this important problem, but two directions stand out as perhaps the major outstanding challenges. One of these is to extend our study to include heterogeneous assemblages of remote computers, whose constituent computers differ in speed and other computational resources. We have already embarked on this study, with an initial report in [30].



(a) Trace 0.

(b) Trace 1.

(c) Trace 2.

(d) Trace 3.

(e) Trace 4.

(f) Trace 5.

(g) Trace 6.

(h) Trace 7.

Fig. 8. Experiments with the eight different traces.

Another great challenge, when the assemblages are heterogeneous, but even when they are homogeneous, is to allow the assemblage's computers to be subject to differing probabilities of failing/being interrupted.

Acknowledgments. The work of A. Benoit and Y. Robert was supported in part by the ANR StochaGrid project. The work of A. Rosenberg was supported in part by US NSF Grants CNS-0842578 and CNS-0905399.

The authors would like to thank Joshua Wingstrom who gave them access to the availability traces.

REFERENCES

[1] A. Benoit, Y. Robert, A. L. Rosenberg, and F. Vivien, "Static strategies for worksharing with unrecoverable interruptions," in *23rd Intl. Parallel and Distributed Processing Symp. (IPDPS)*. IEEE Computer Society Press, 2009.

[2] G. F. Pfister, *In Search of Clusters*. Prentice-Hall, 1995.

[3] R. Buyya, D. Abramson, and J. Giddy, "A case for economy grid architecture for service-oriented grid computing," in *10th Heterogeneous Computing Workshop*. IEEE Computer Society, 2001.

[4] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Intl. J. High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[5] I. Foster and C. Kesselman, Eds., *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 2004.

[6] W. Cirne and K. Marzullo, "The computational co-op: Gathering clusters into a metacomputer," in *13th Intl. Parallel Processing Symp. (IPPS)*, 1999, pp. 160–166.

[7] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky, "Seti@home-massively distributed computing for seti," *Computing in Science & Engineering*, vol. 3, no. 1, pp. 78–83, 2001.

[8] B. Awerbuch, Y. Azar, A. Fiat, and F. T. Leighton, "Making commitments in the face of uncertainty: How to pick a winner almost every time," in *28th ACM SToC*, 1996, pp. 519–530.

[9] M. A. Bender and C. A. Phillips, "Scheduling dags on asynchronous processors," in *19th ACM SPAA*, 2007, pp. 35–45.

[10] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, "An optimal strategies for cycle-stealing in networks of workstations," *IEEE Trans. Computers*, vol. 46, no. 5, pp. 545–557, 1997.

[11] L. Gao and G. Malewicz, "Toward maximizing the quality of results of dependent tasks computed unreliably," *Theory Comput. Syst.*, vol. 41, no. 4, pp. 731–752, 2007.

[12] G. Malewicz, A. L. Rosenberg, and M. Yurkewych, "Toward a theory for scheduling dags in internet-based computing," *IEEE Trans. Computers*, vol. 55, no. 6, pp. 757–768, 2006.

[13] A. L. Rosenberg, "Optimal schedules for cycle-stealing in a network of workstations with a bag-of-tasks workload," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 2, pp. 179–191, 2002.

[14] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems*. Wiley-IEEE Computer Society Press, 1996.

[15] A. L. Rosenberg, "Changing challenges for collaborative algorithmics," in *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*, A. Zomaya, Ed. Springer, 2006, pp. 1–44.

[16] ——, "Guidelines for data-parallel cycle-stealing in networks of workstations i: On maximizing expected output," *J. Parallel Distrib. Comput.*, vol. 59, no. 1, pp. 31–53, 1999.

[17] ——, "Guidelines for data-parallel cycle-stealing in networks of workstations ii: On maximizing guaranteed output," *Intl. J. Foundations of Computer Science*, vol. 11, no. 1, pp. 183–204, 2000.

[18] D. Kondo, H. Casanova, E. Wing, and F. Berman, "Models and scheduling mechanisms for global computing applications," in *16th Intl. Parallel and Distr. Processing Symp. (IPDPS)*, 2002.

[19] J. Wingstrom and H. Casanova, "Probabilistic allocation of tasks on desktop grids," in *Proceedings of PCGrid*. IEEE CS Press, 2008.

[20] C. Anglano, J. Brevik, M. Canonico, D. Nurmi, and R. Wolski, "Fault-aware scheduling for bag-of-tasks applications on desktop grids," in *GRID '06*. IEEE Computer Society, 2006, pp. 56–63.

[21] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang, "Scheduling divisible loads on star and tree networks: results and open problems," *IEEE TPDS*, vol. 16, no. 3, pp. 207–218, 2005.

[22] M. Gallet, Y. Robert, and F. Vivien, "Divisible load scheduling," in *Introduction to Scheduling*. Chapman and Hall/CRC Press, 2009.

[23] ——, "Comments on "design and performance evaluation of load distribution strategies for multiple loads on heterogeneous linear daisy chain networks"," *J. Parallel Distributed Computing*, vol. 68, no. 7, pp. 1021–1031, 2008.

[24] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor - a hunter of idle workstations," in *ICDCS*, 1988, pp. 104–111.

[25] S. White and D. Torney, "Use of a workstation cluster for the physical mapping of chromosomes," *SIAM NEWS*, pp. 14–17, Mar. 1993.

[26] D. Kondo, "Scheduling task parallel applications for rapid turnaround on enterprise desktop grids," Ph.D. dissertation, University of California at San Diego, July 2005.

[27] R. H. Arpaci, A. C. Dusseau, A. M. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson, "The interaction of parallel and sequential workloads on a network of workstations," in *SIGMETRICS '95/PERFORMANCE '95*. ACM, 1995, pp. 267–278.

[28] J. Plank and W. Elwasif, "Experimental assessment of workstation failures and their impact on checkpointing systems," in *Fault-Tolerant Computing, 1998*, June 1998, pp. 48–57.

[29] D. Nurmi, J. Brevik, and R. Wolski, "Modeling machine availability in enterprise and wide-area distributed computing environments," in *Euro-Par 2005*, vol. LNCS 3648, 2005, pp. 432–441.

[30] A. Benoit, Y. Robert, A. Rosenberg, and F. Vivien, "Static worksharing strategies for heterogeneous computers with unrecoverable interruptions," *Parallel Computing*, 2010, to appear.

APPENDIX A
EXPERIMENTS WITH LINEAR RISK FUNCTIONS (SELECTED HEURISTICS)

On the following graphs, the only group-heuristic whose performance is reported is Σ_{greedy}.

A.1 Experiments E1

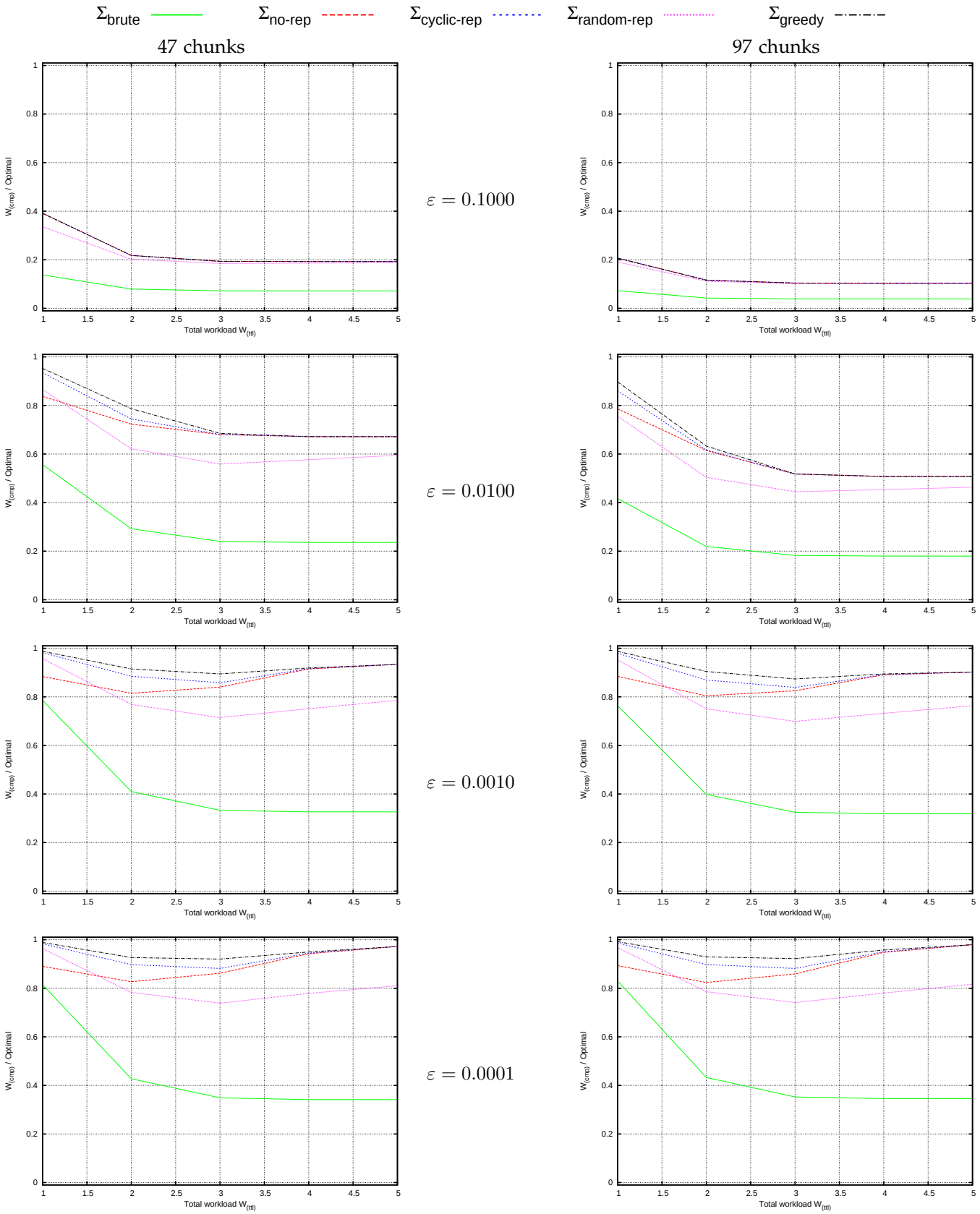


Fig. 10. Experiment (E1) using 5 computers.

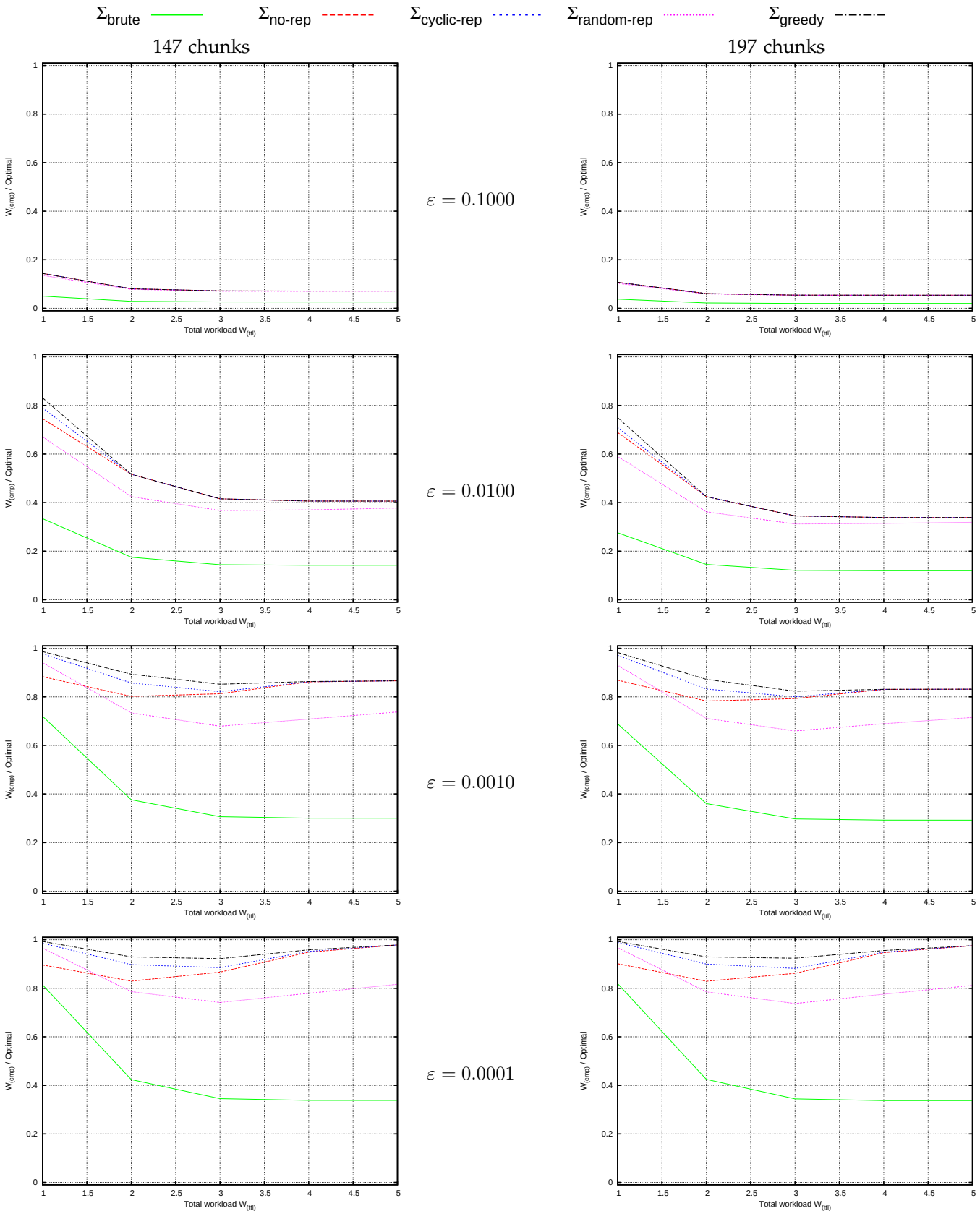


Fig. 11. Experiment (E1) using 5 computers (continued).

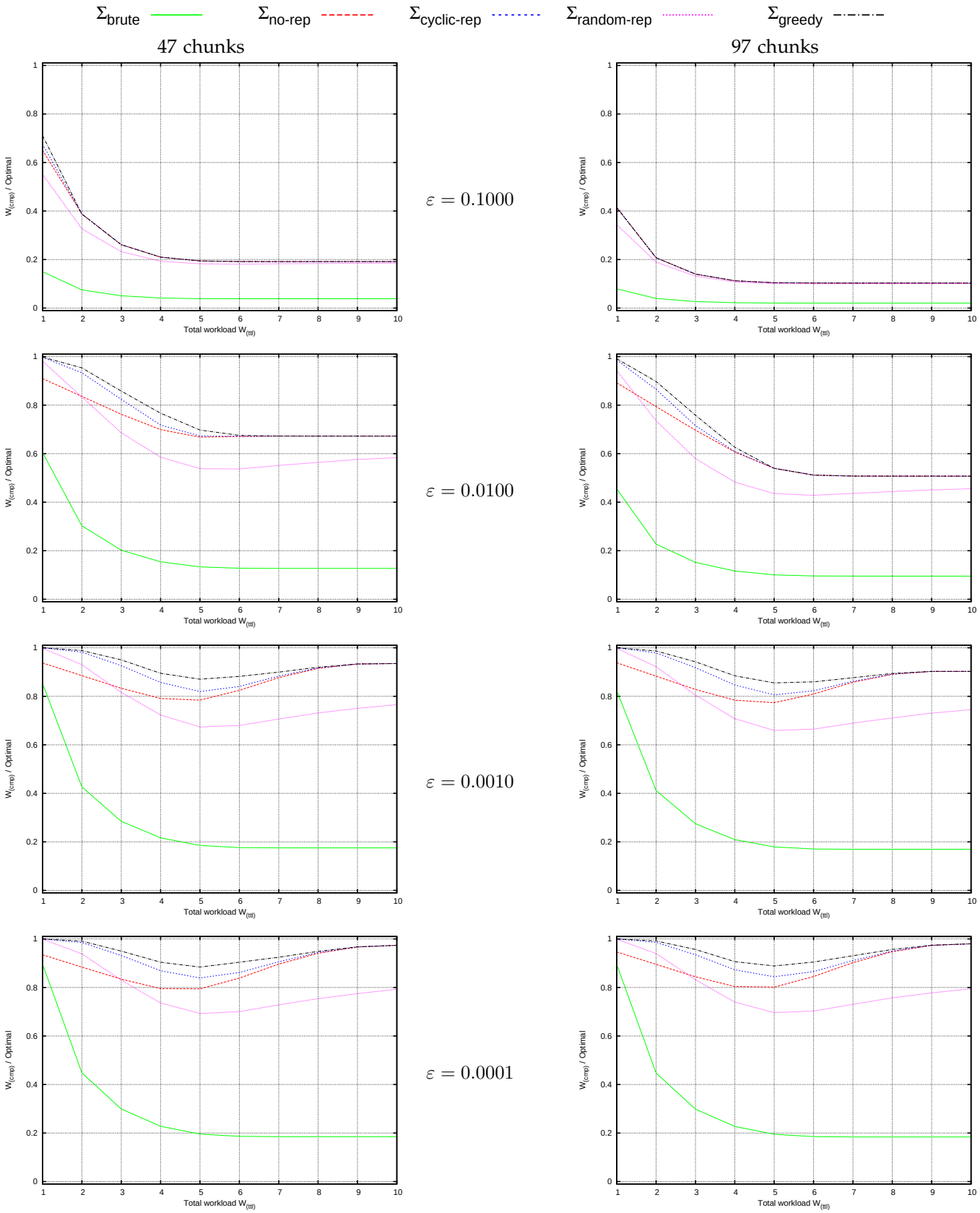


Fig. 12. Experiment (E1) using 10 computers.

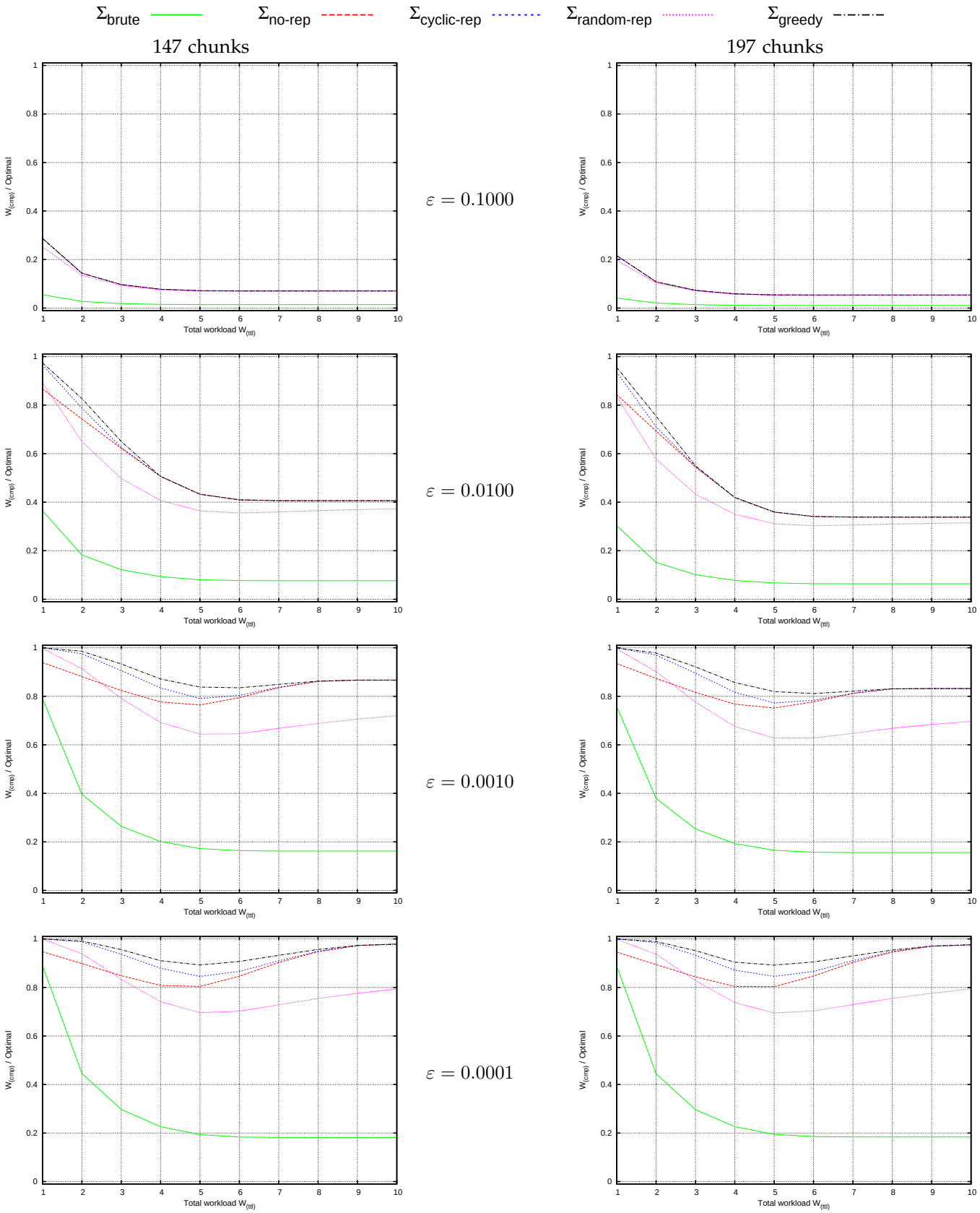


Fig. 13. Experiment (E1) using 10 computers (continued).

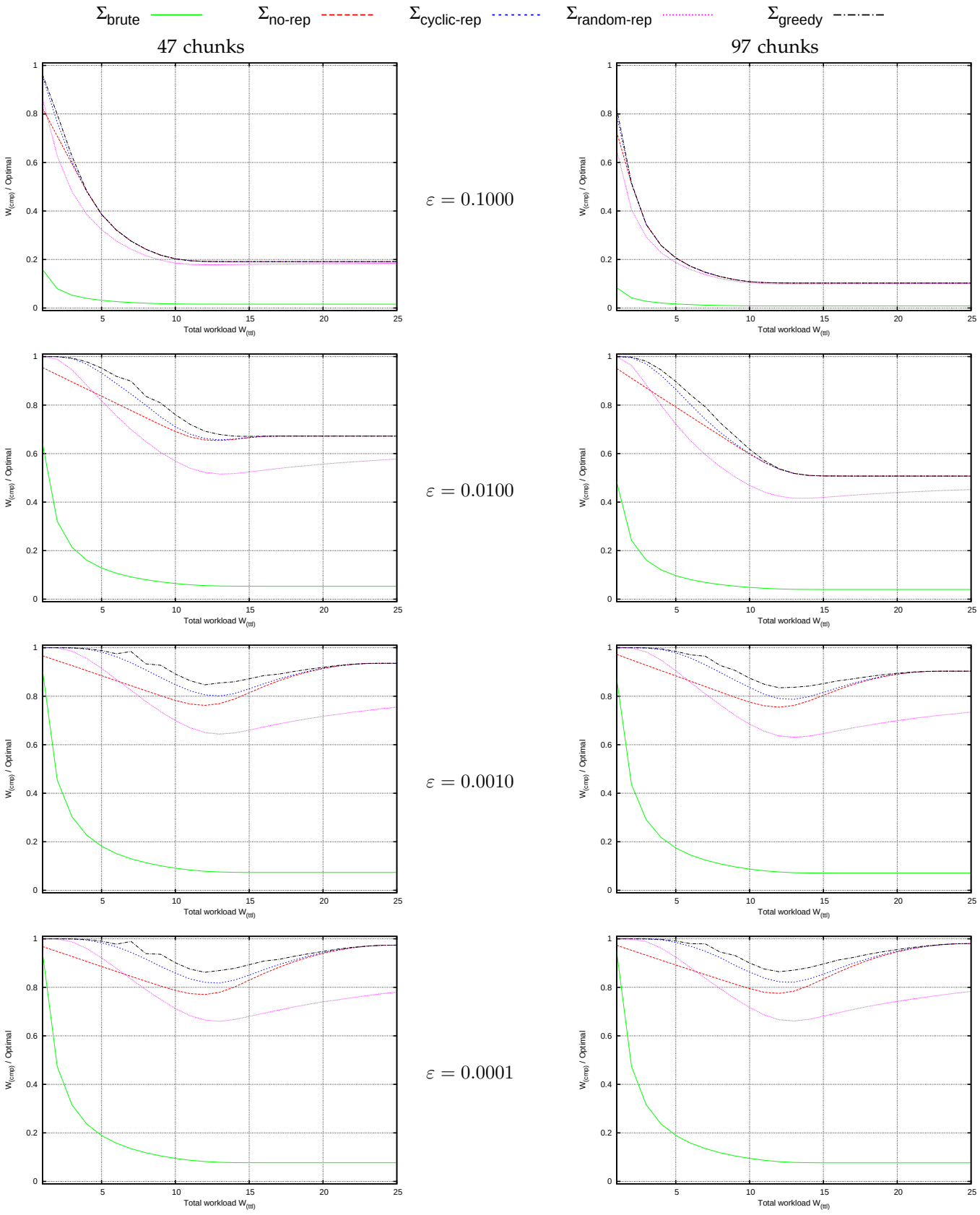


Fig. 14. Experiment (E1) using 25 computers.

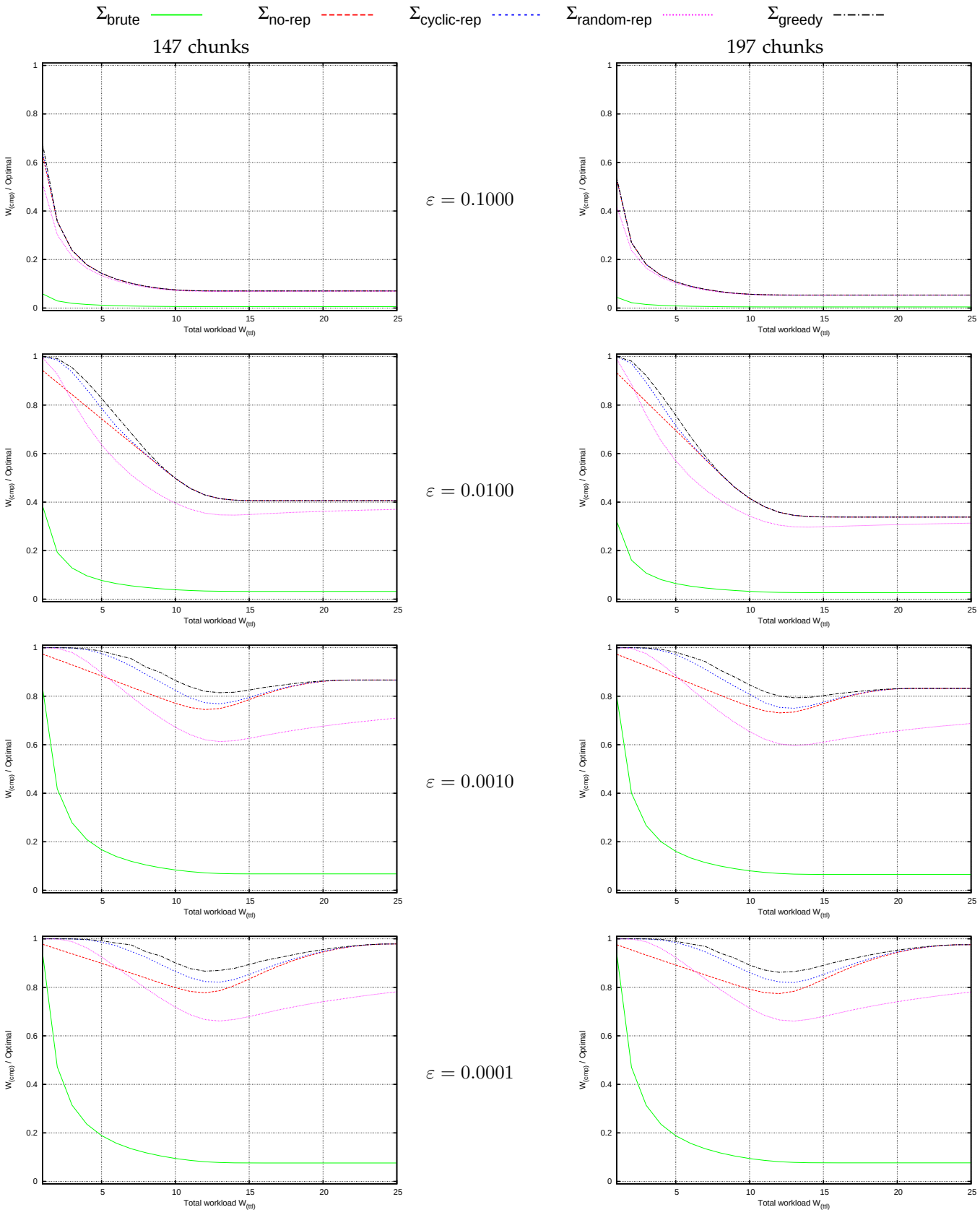


Fig. 15. Experiment (E1) using 25 computers (continued).

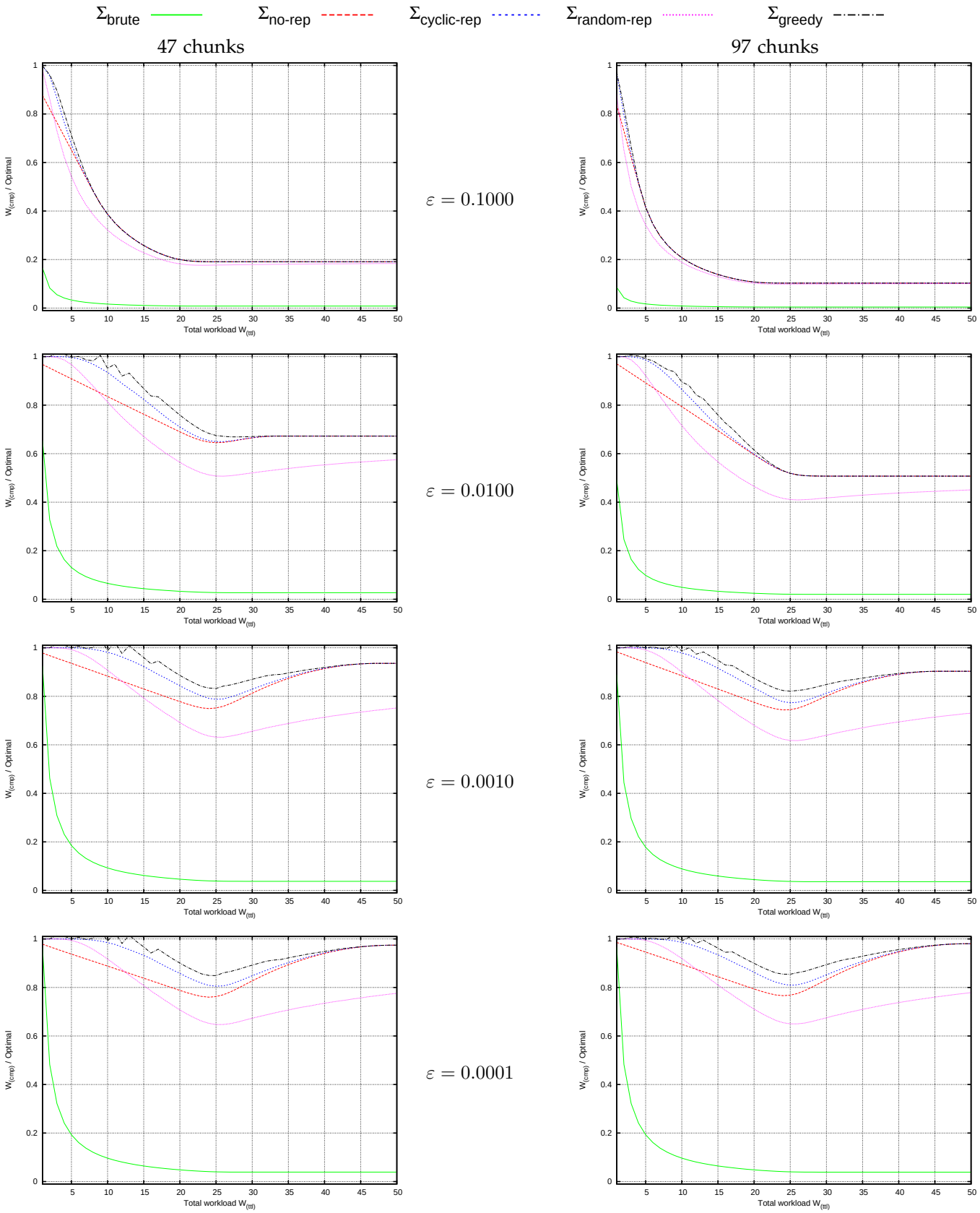


Fig. 16. Experiment (E1) using 50 computers.

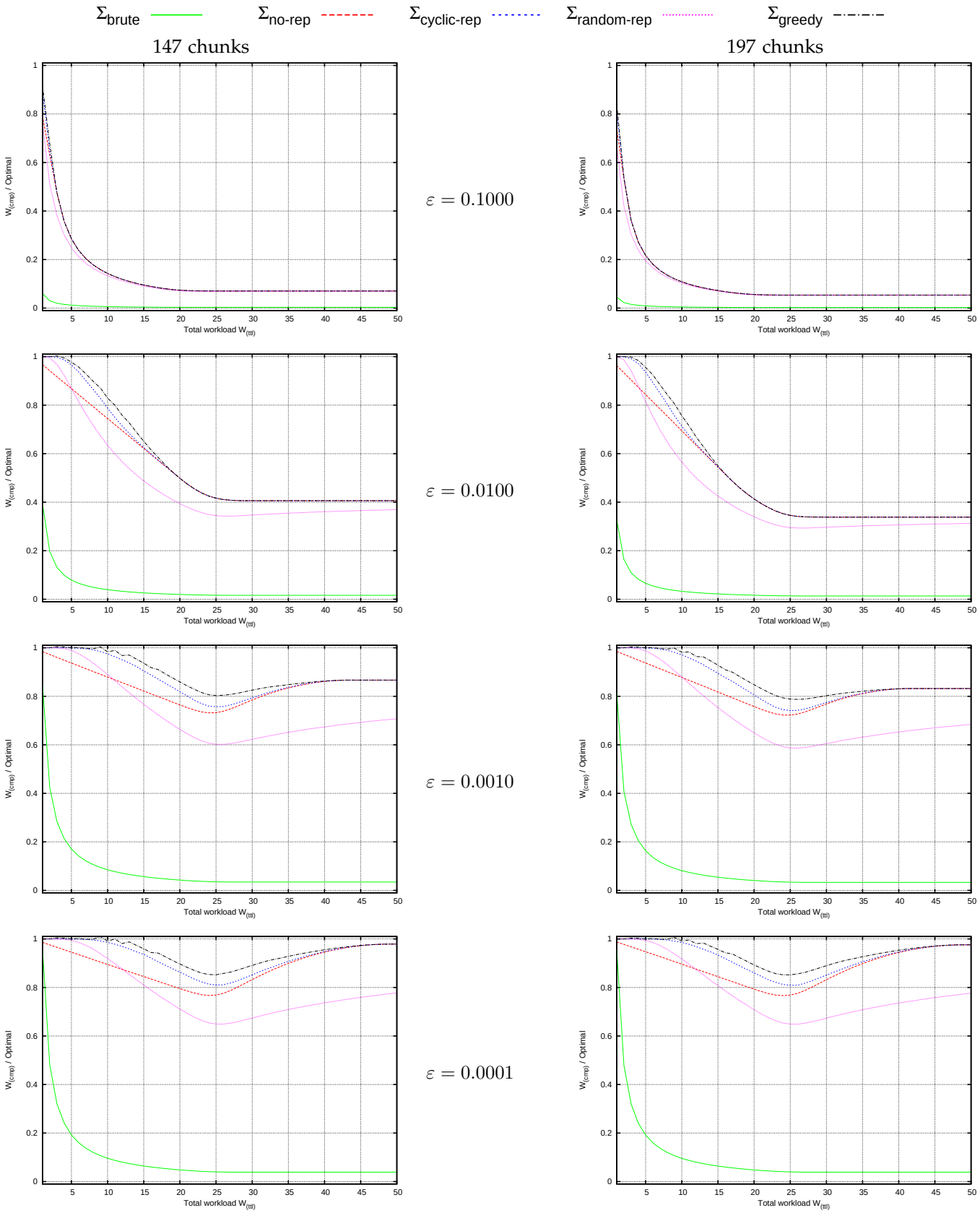


Fig. 17. Experiment (E1) using 50 computers (continued).

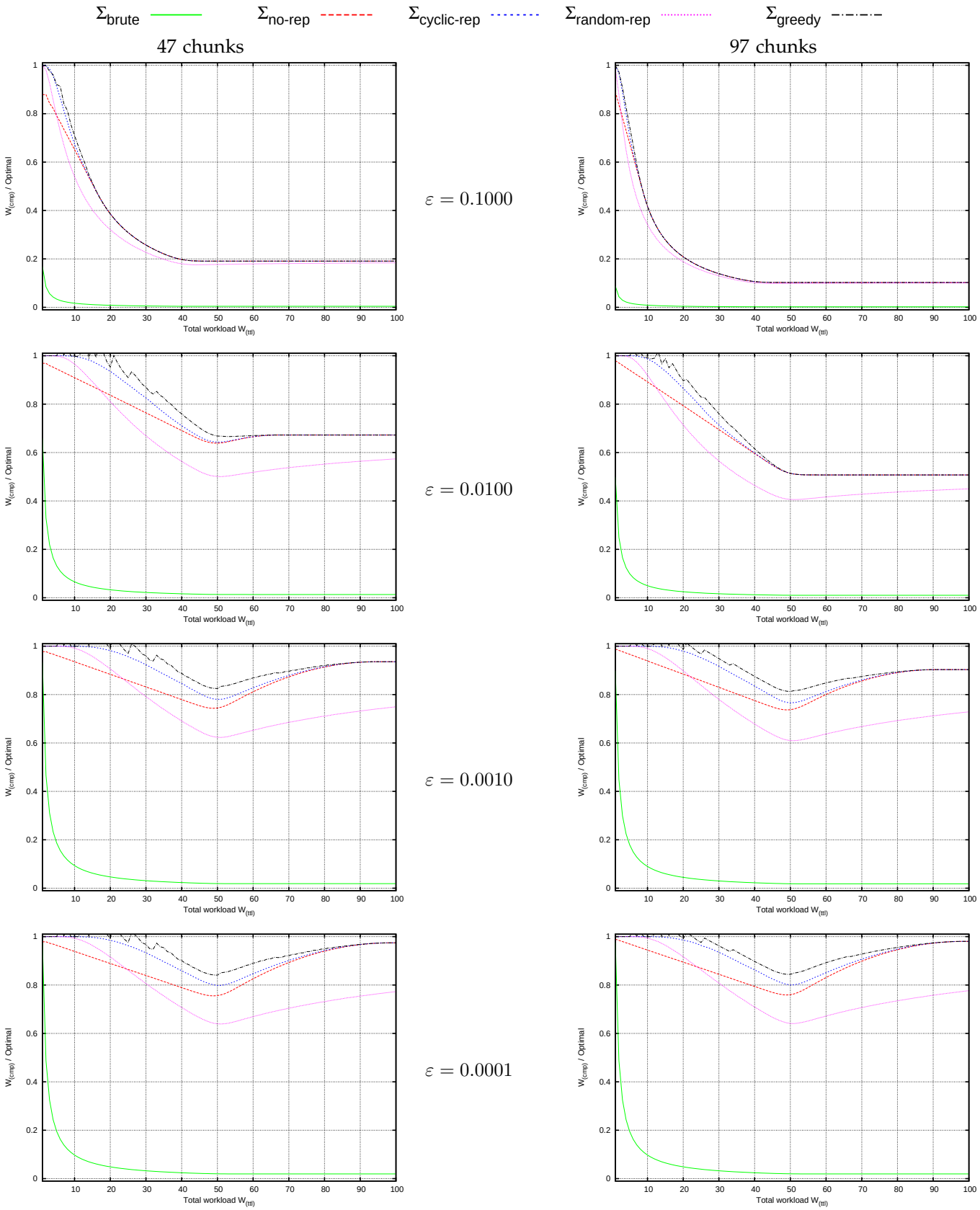


Fig. 18. Experiment (E1) using 100 computers.

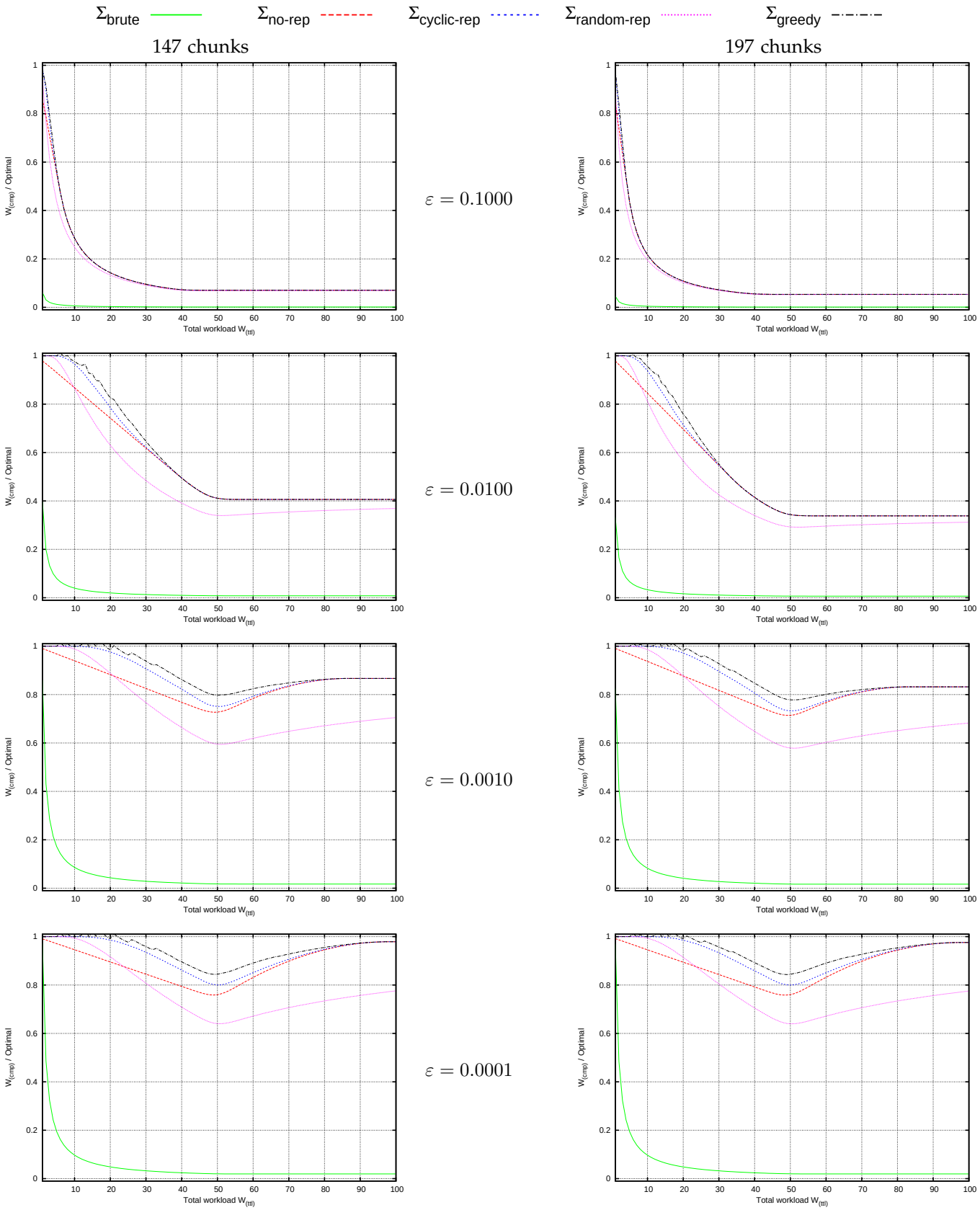


Fig. 19. Experiment (E1) using 100 computers (continued).

A.2 Experiments E2

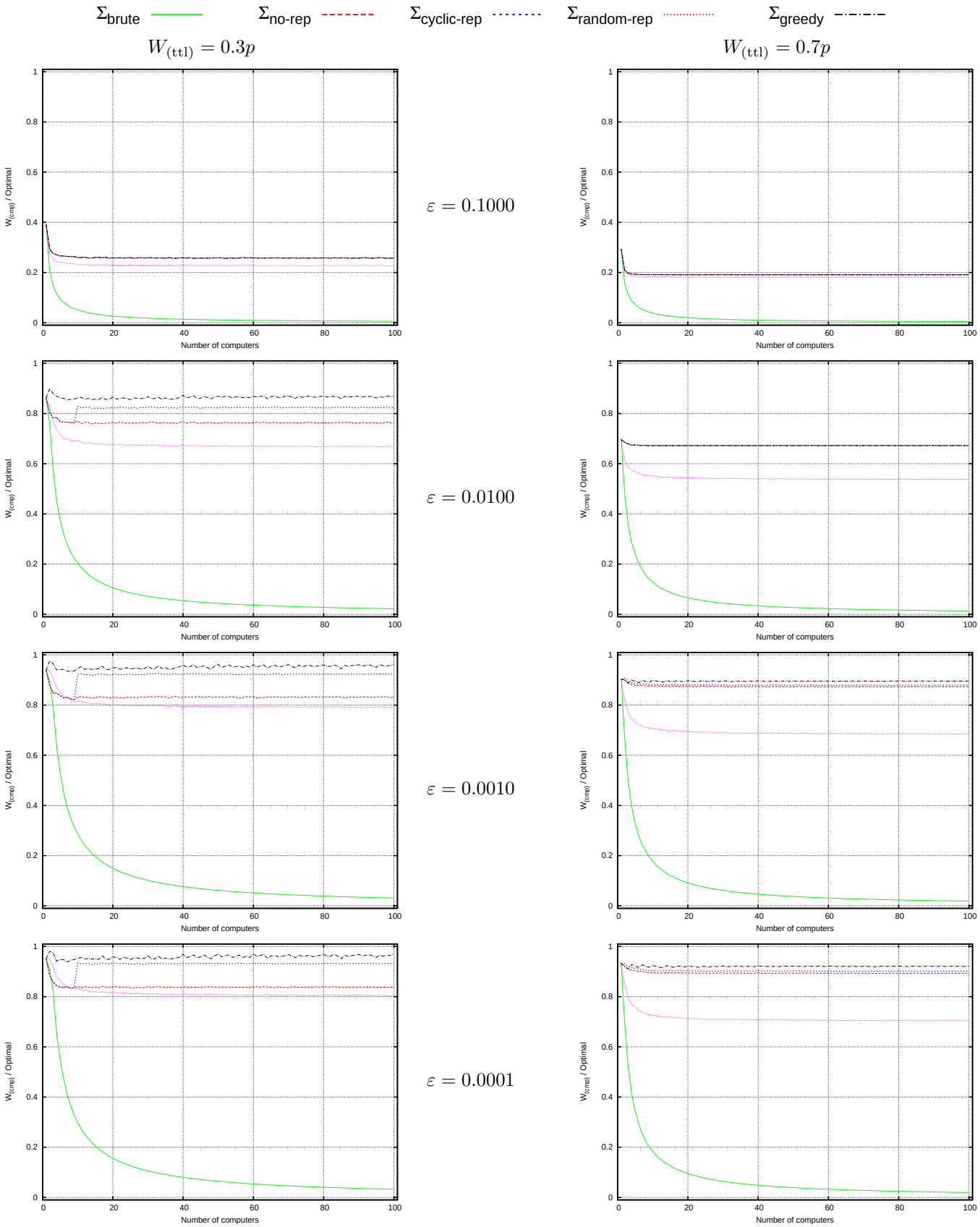


Fig. 20. Experiment (E2) with 47 chunks.

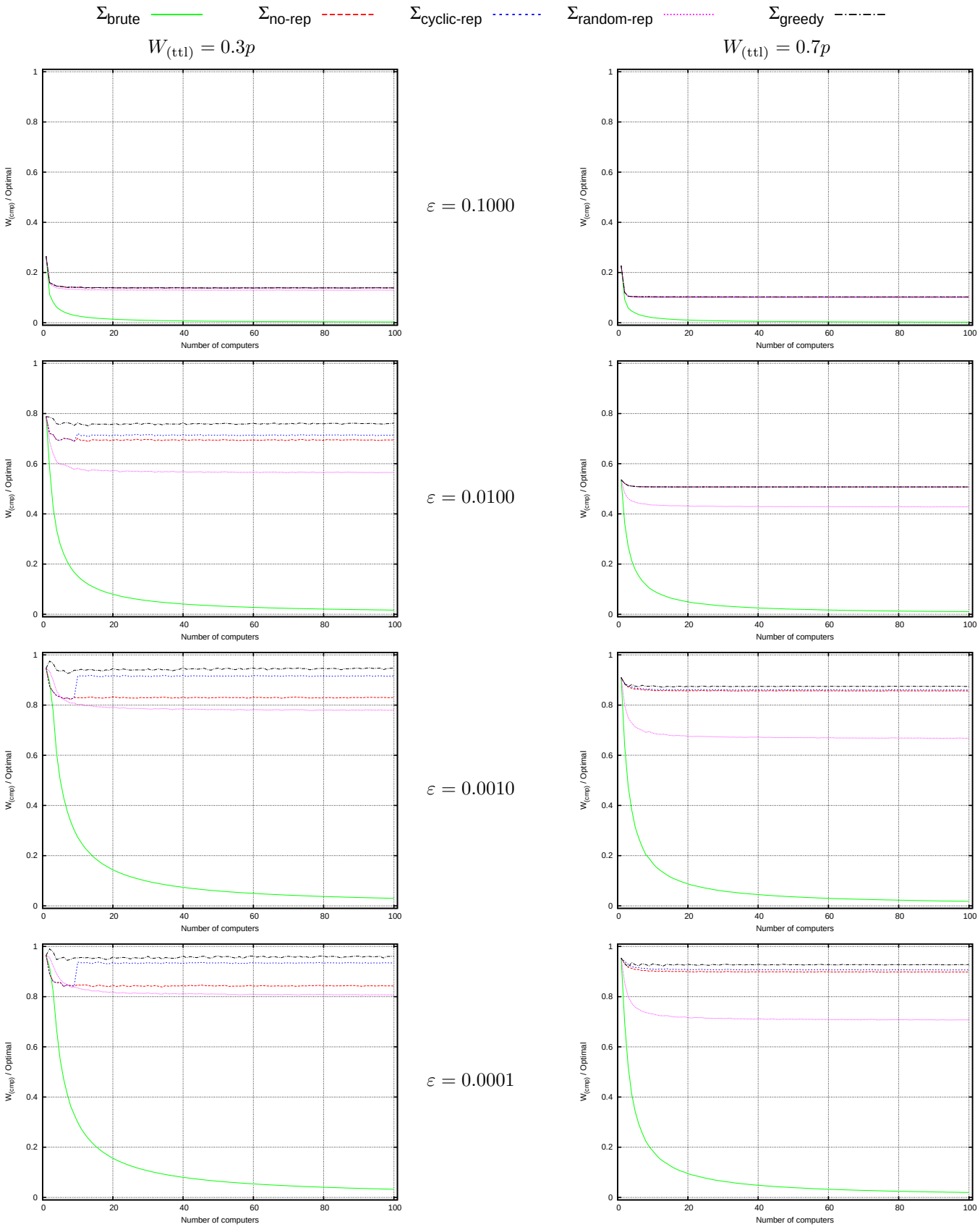


Fig. 21. Experiment (E2) with 97 chunks.

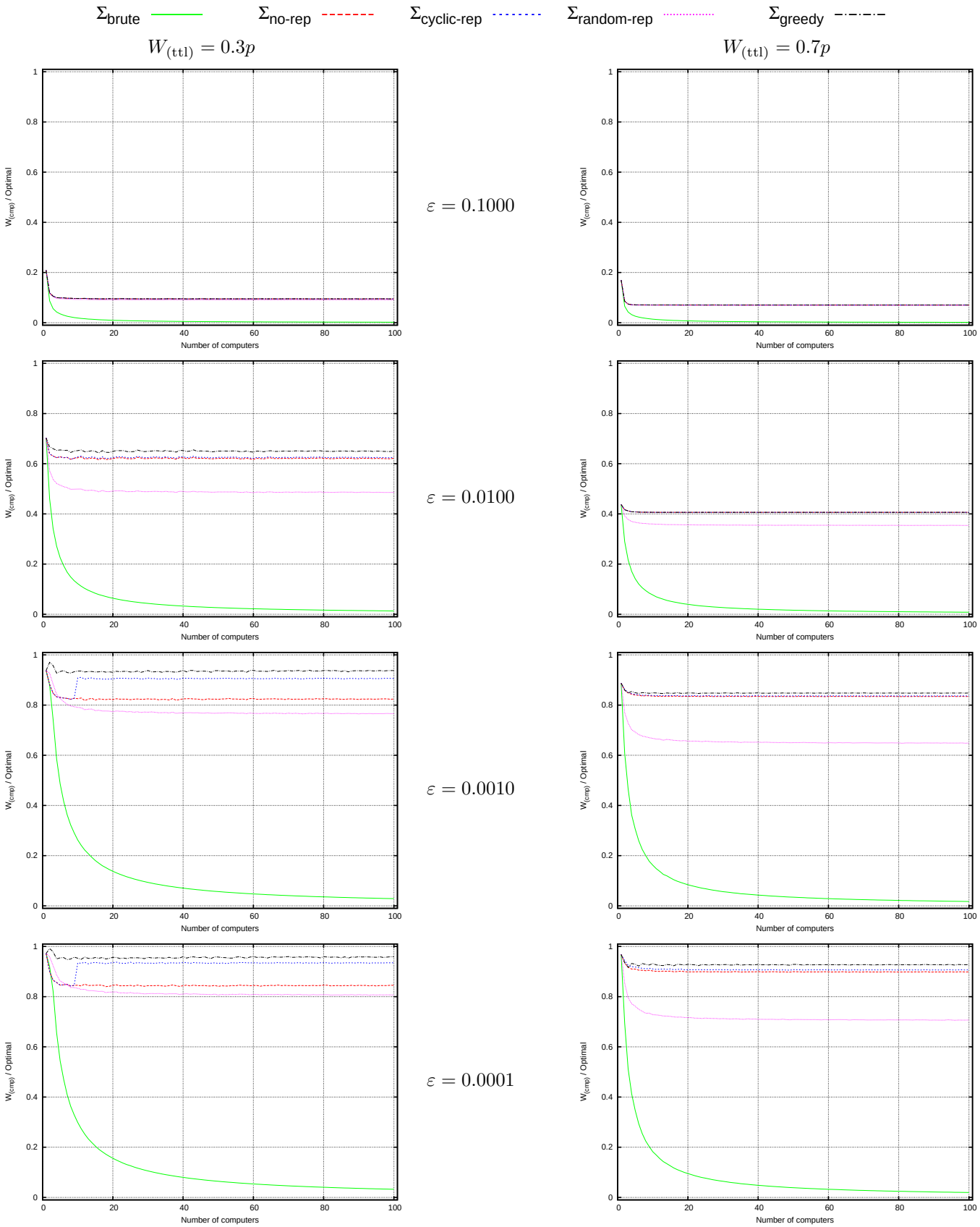


Fig. 22. Experiment (E2) with 147 chunks.

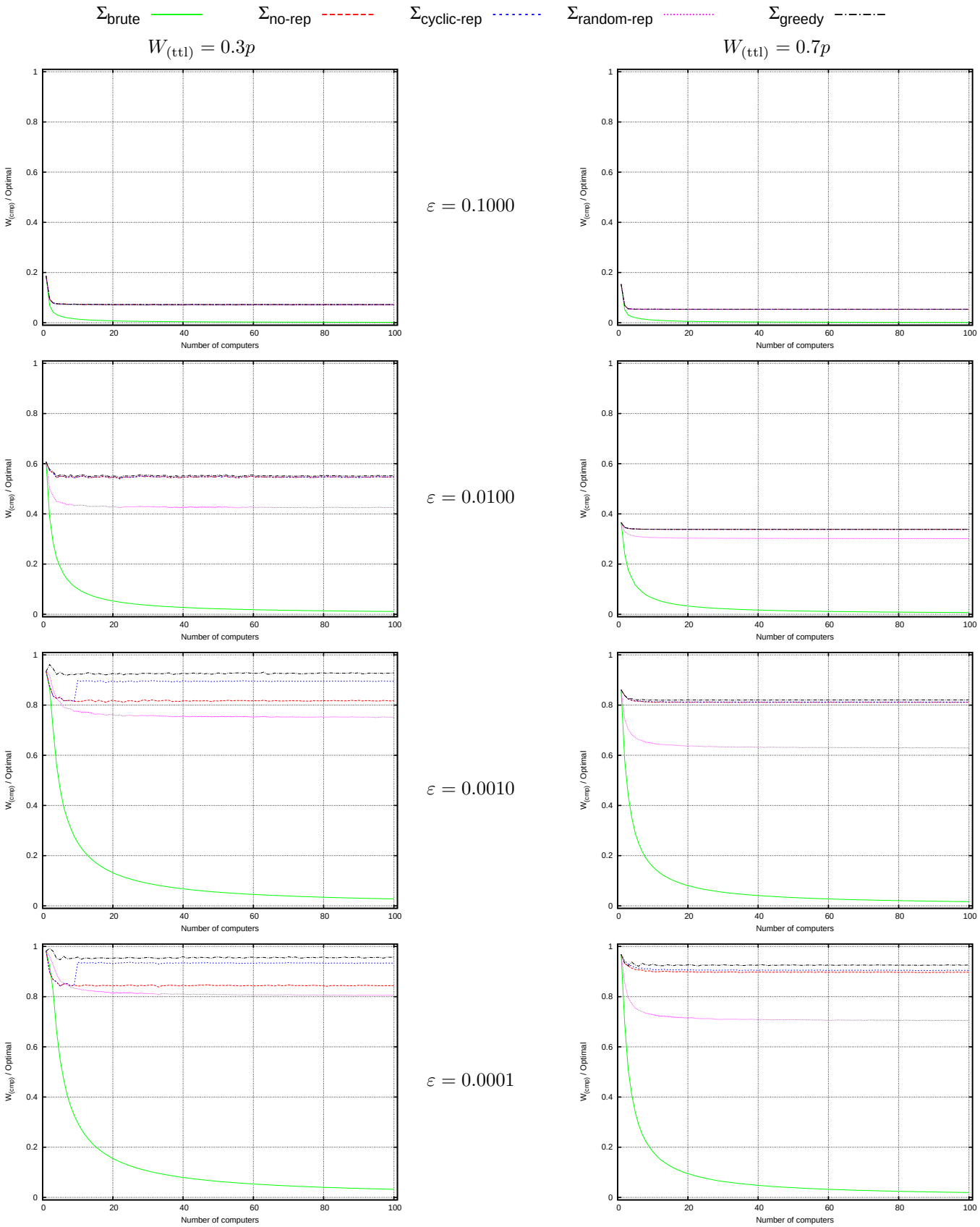


Fig. 23. Experiment (E2) with 197 chunks.

A.3 Experiments E3

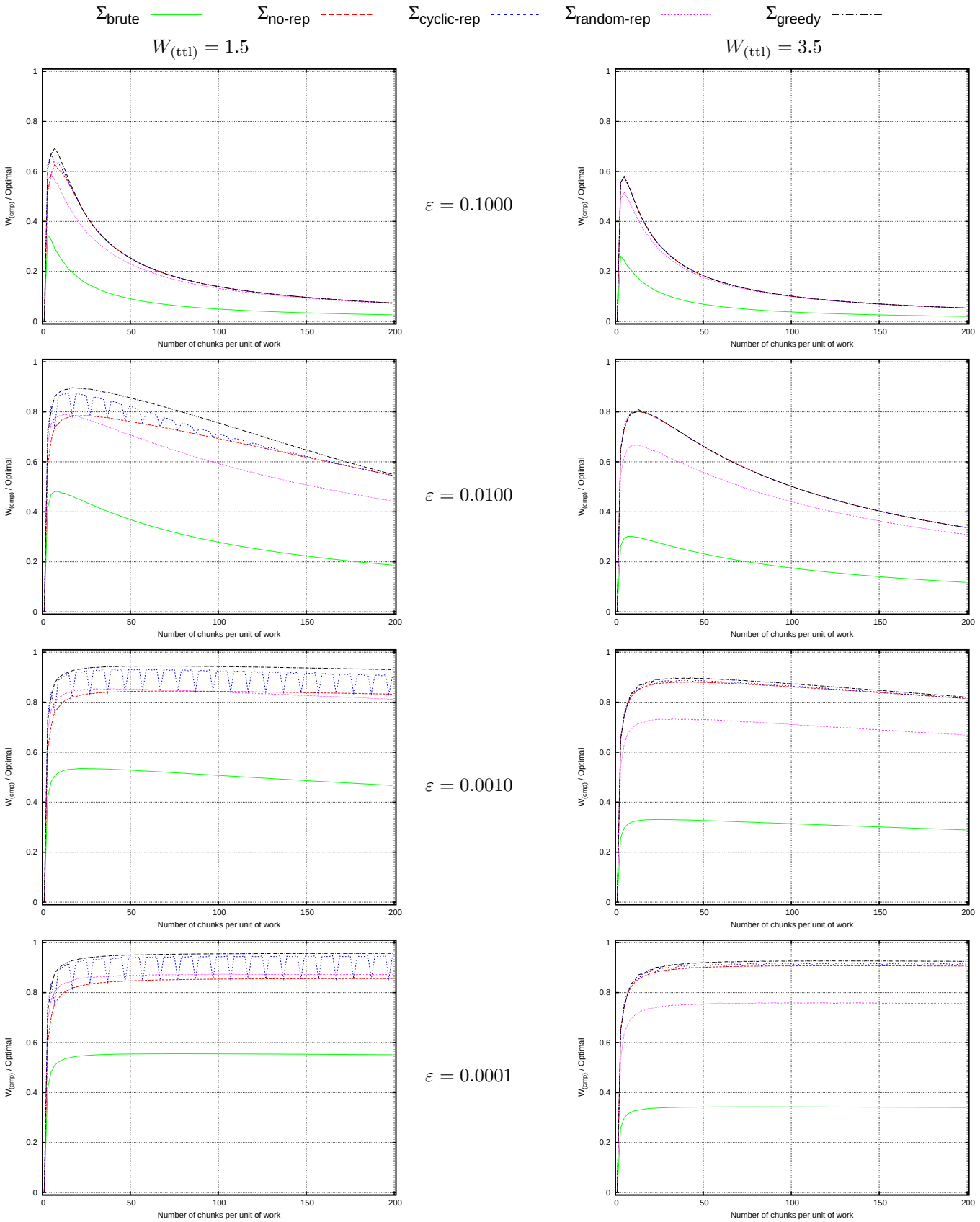


Fig. 24. Experiment (E3) using 5 computers.

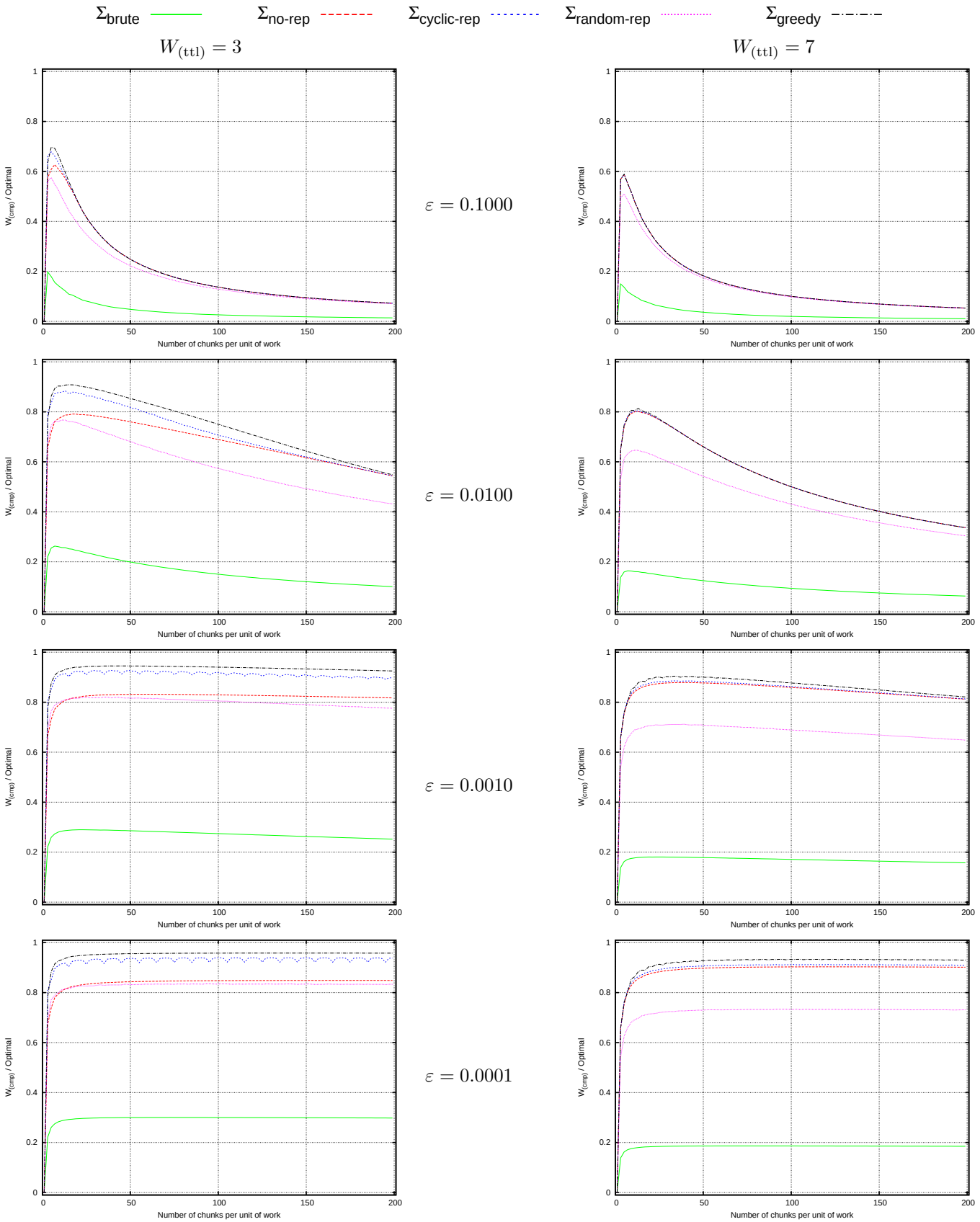


Fig. 25. Experiment (E3) using 10 computers.

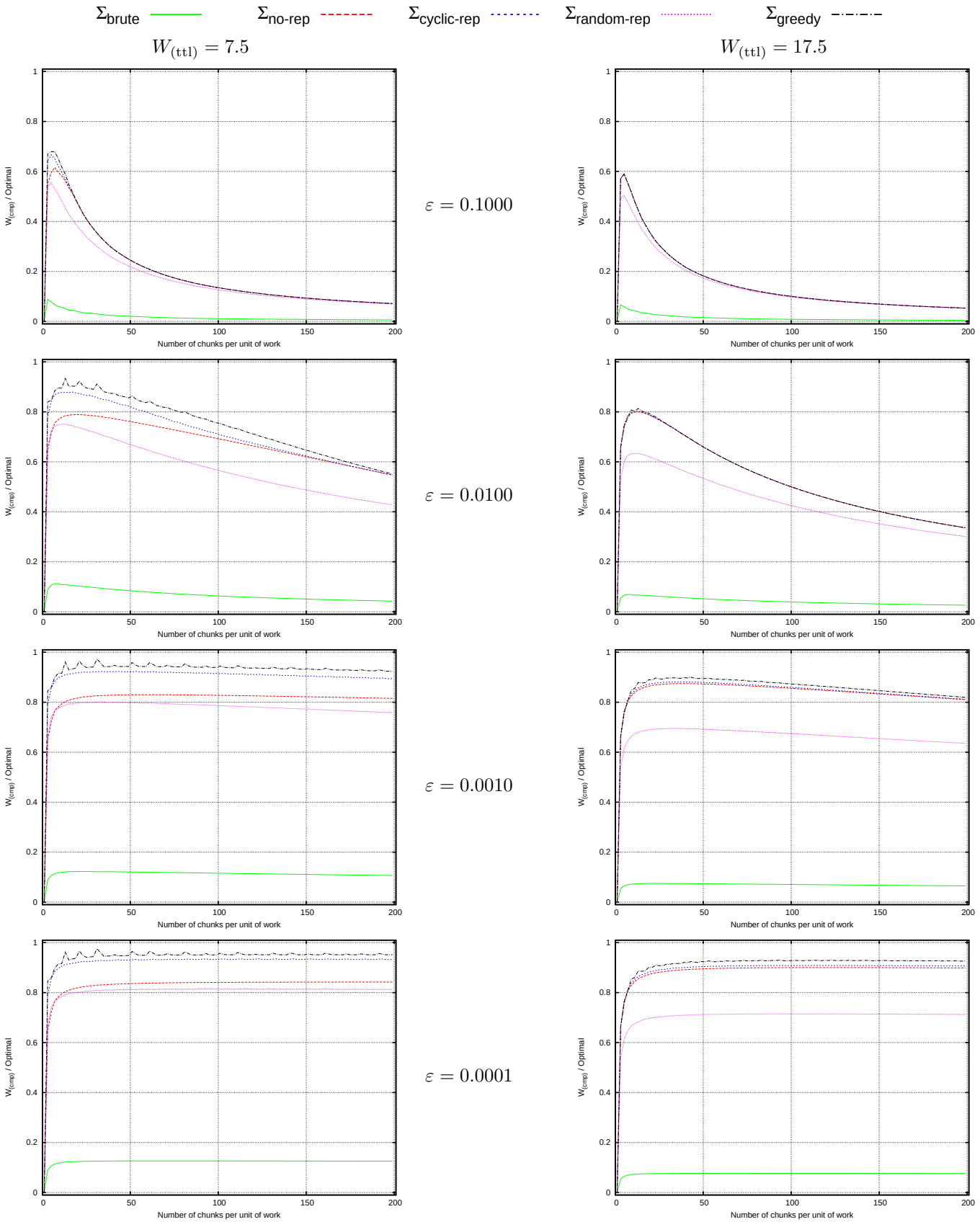


Fig. 26. Experiment (E3) using 25 computers.

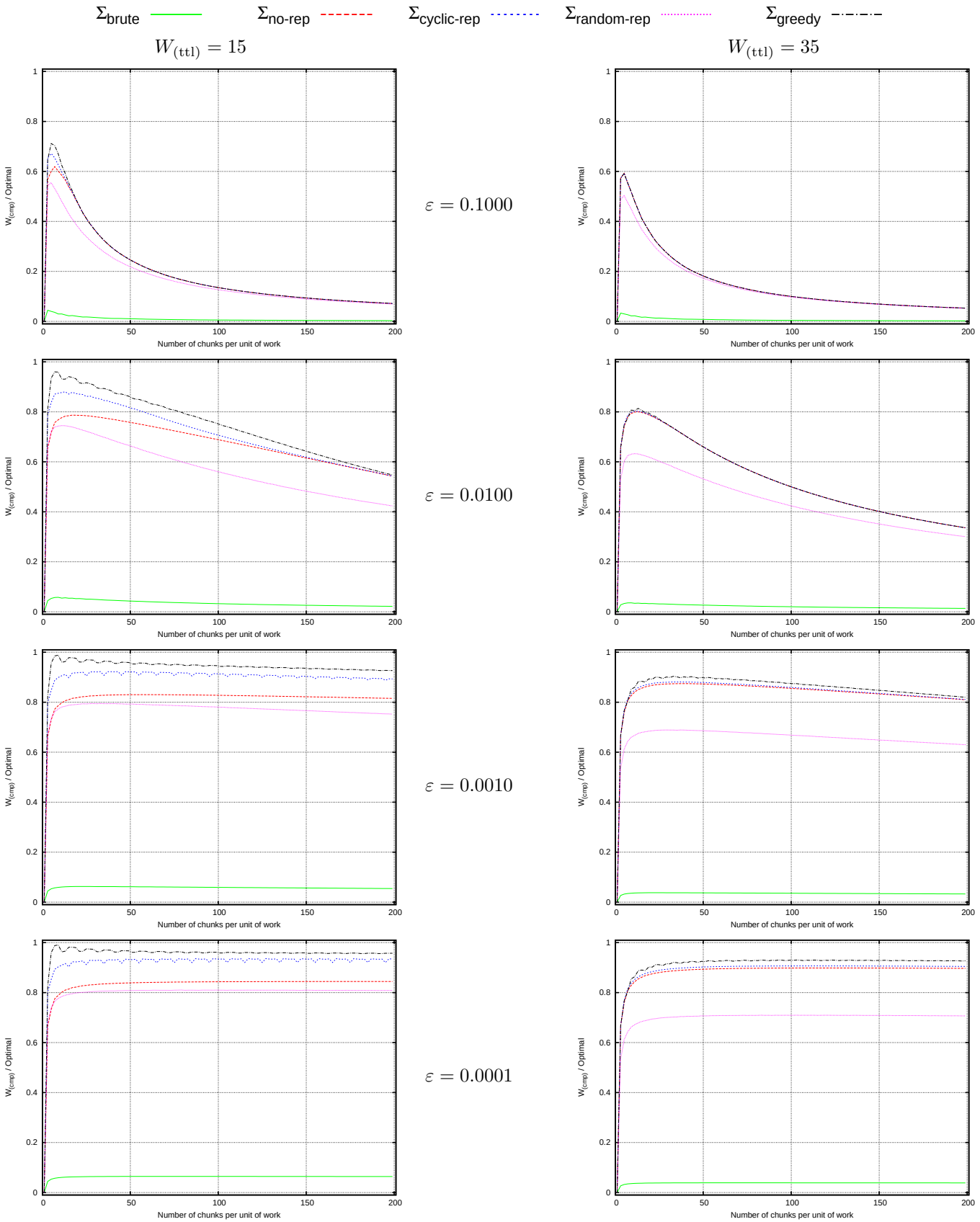


Fig. 27. Experiment (E3) using 50 computers.

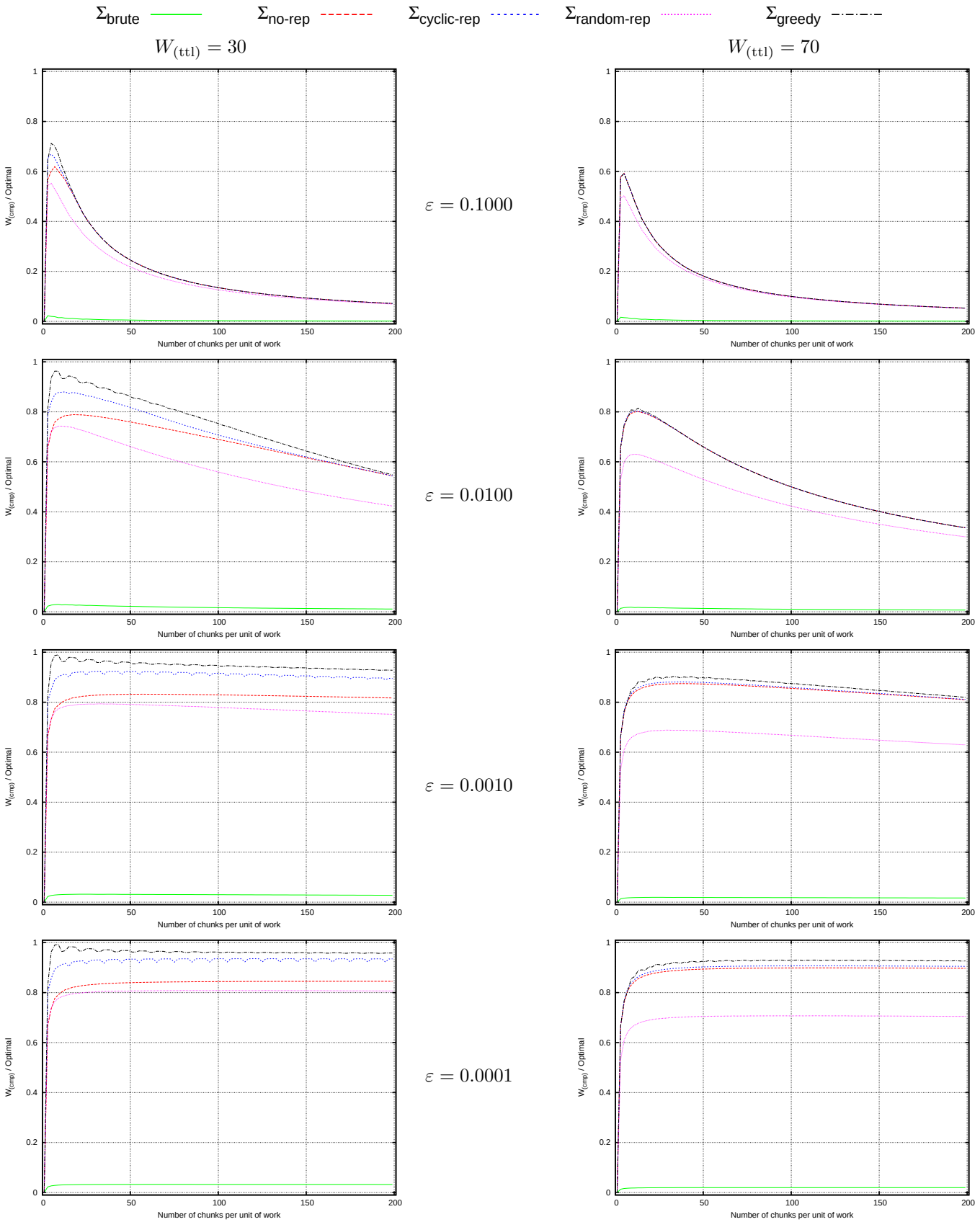


Fig. 28. Experiment (E3) using 100 computers.

A.4 Experiments E4

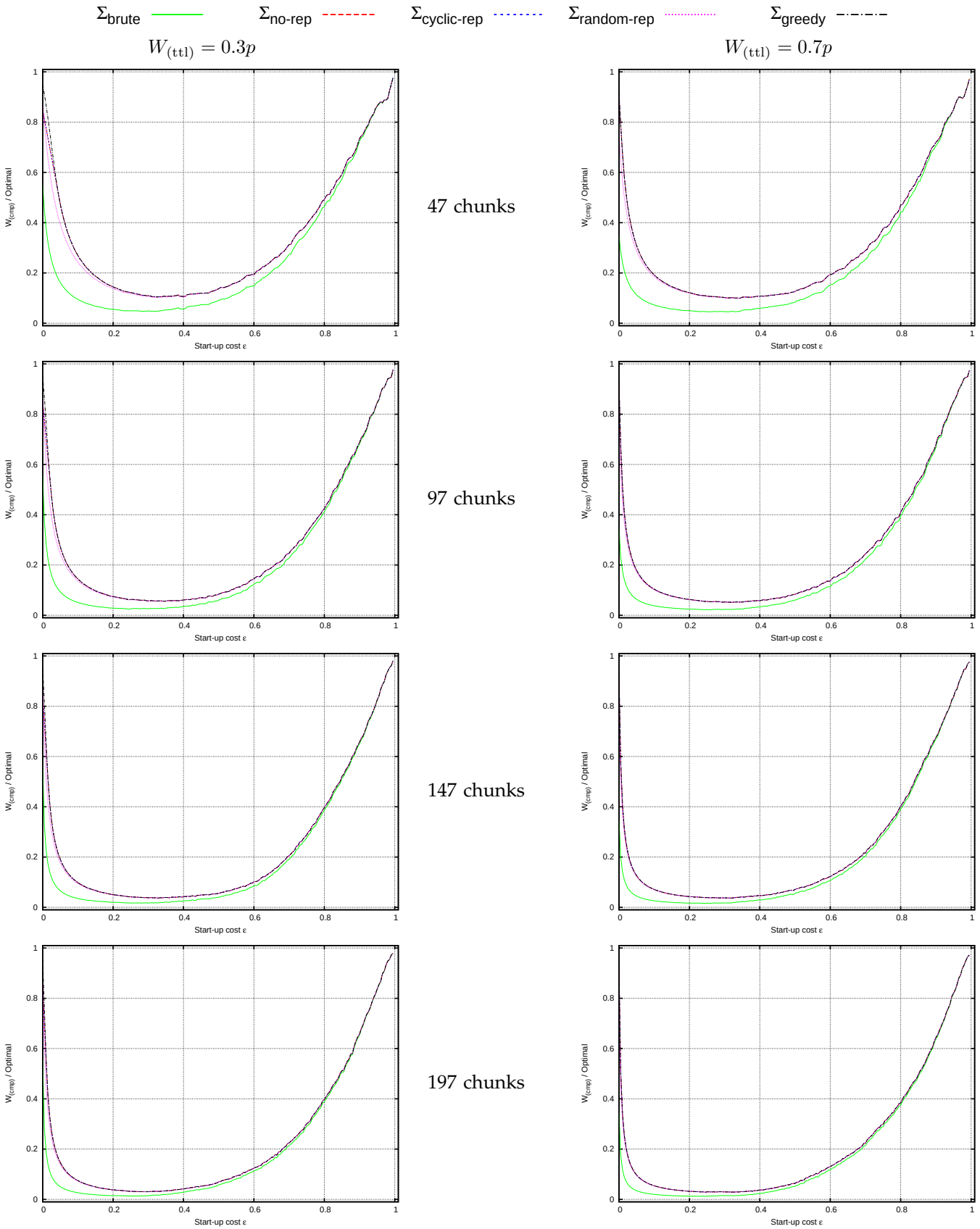


Fig. 29. Experiment (E4) with 5 computers.

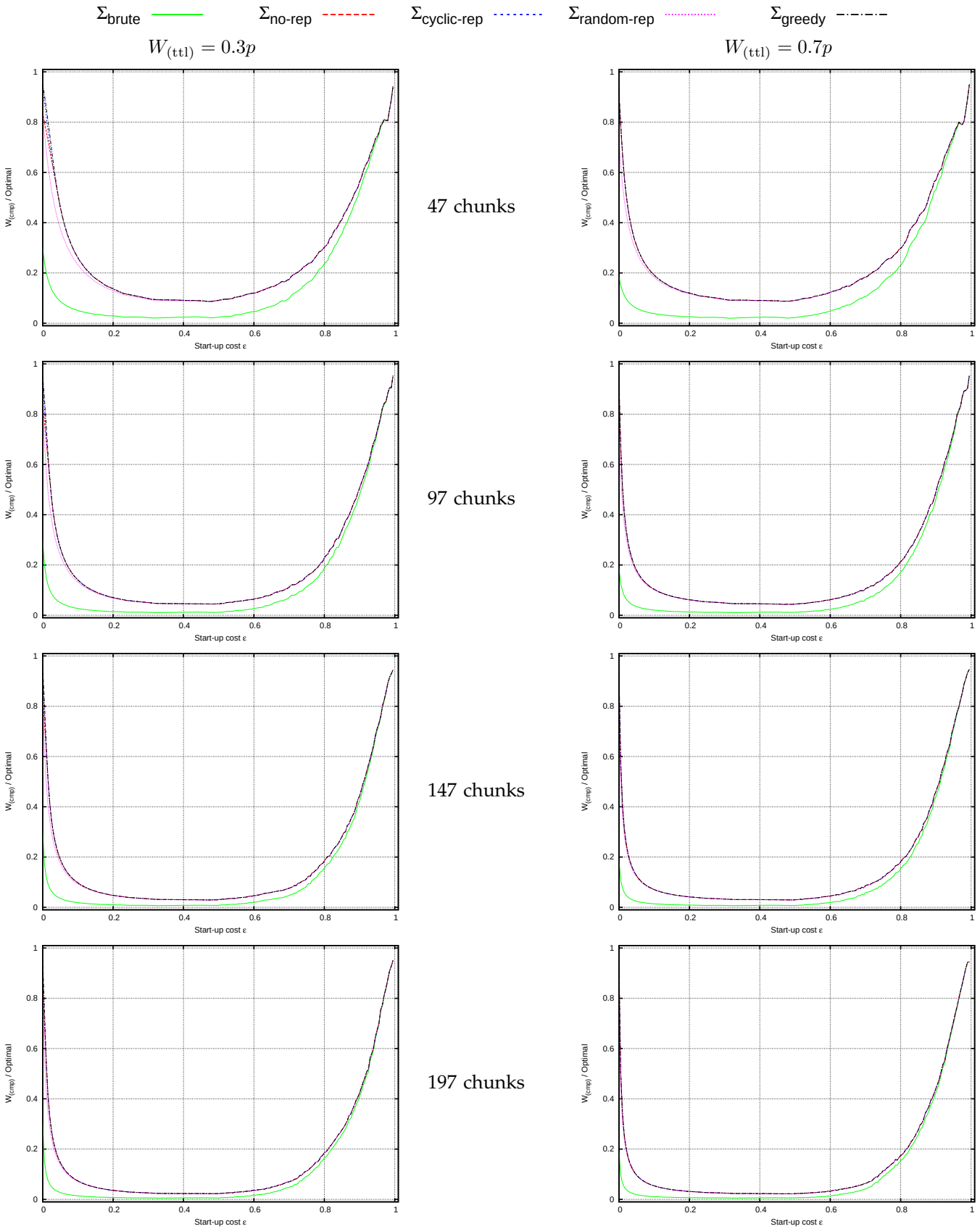


Fig. 30. Experiment (E4) with 10 computers.

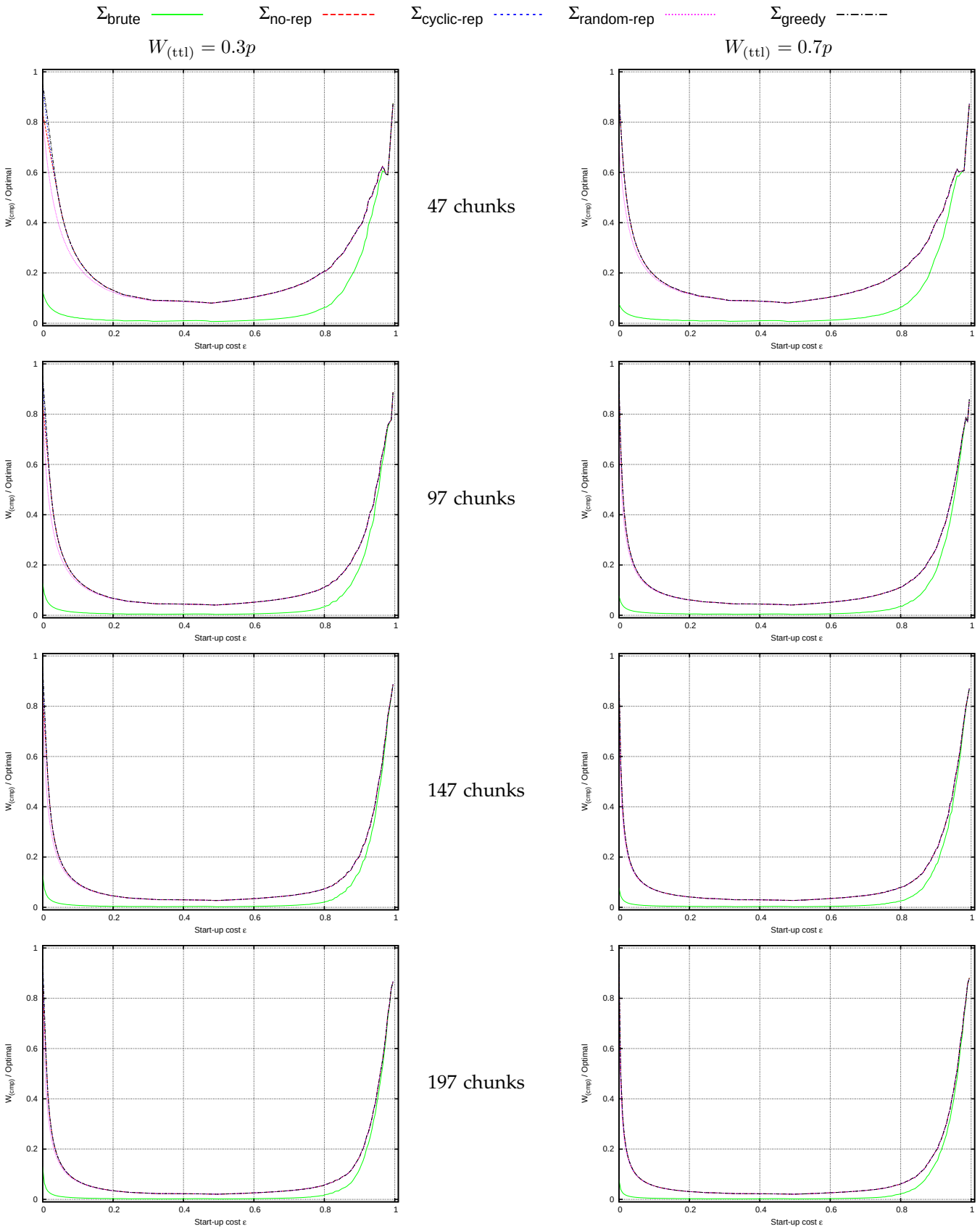


Fig. 31. Experiment (E4) with 25 computers.

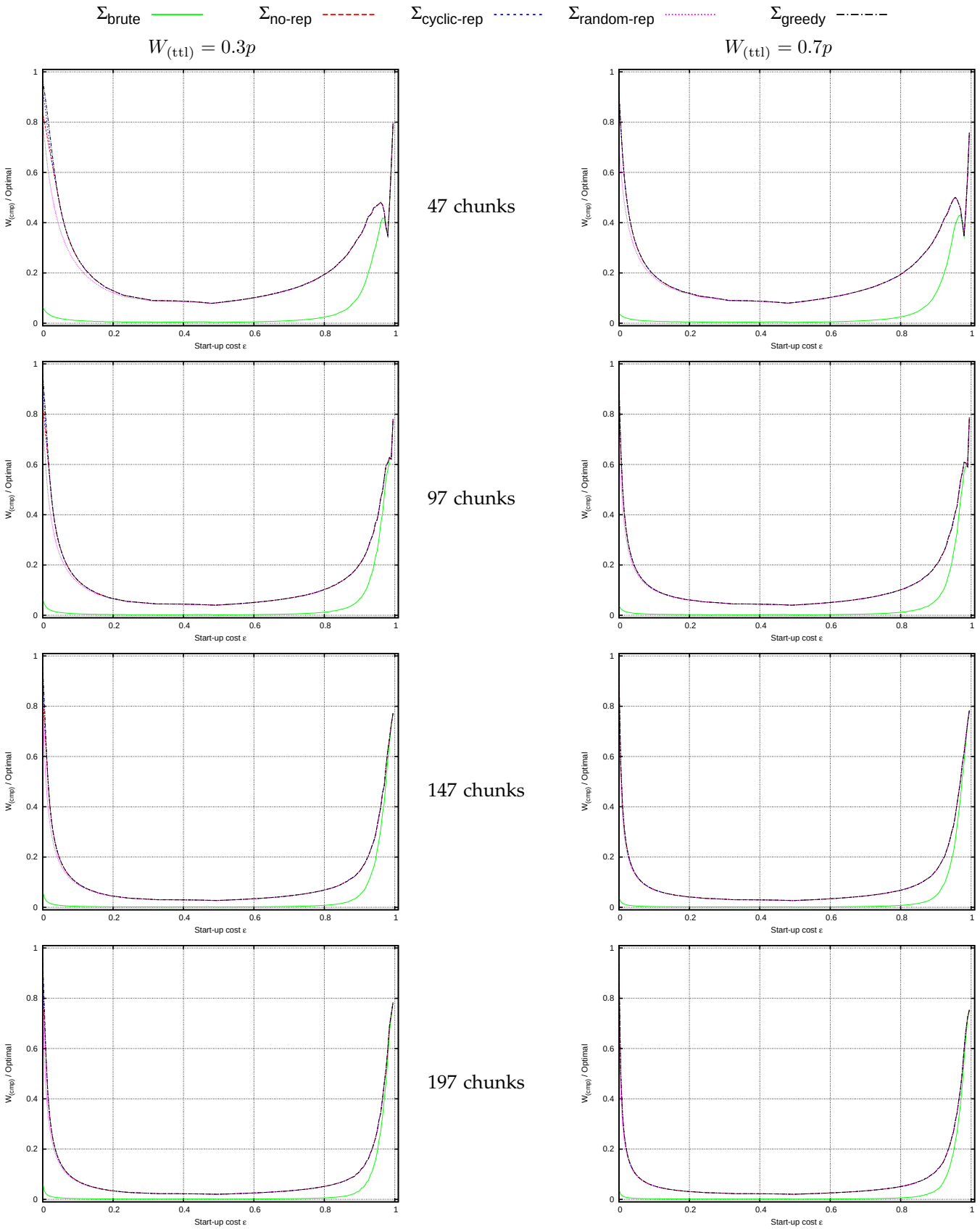


Fig. 32. Experiment (E4) with 50 computers.

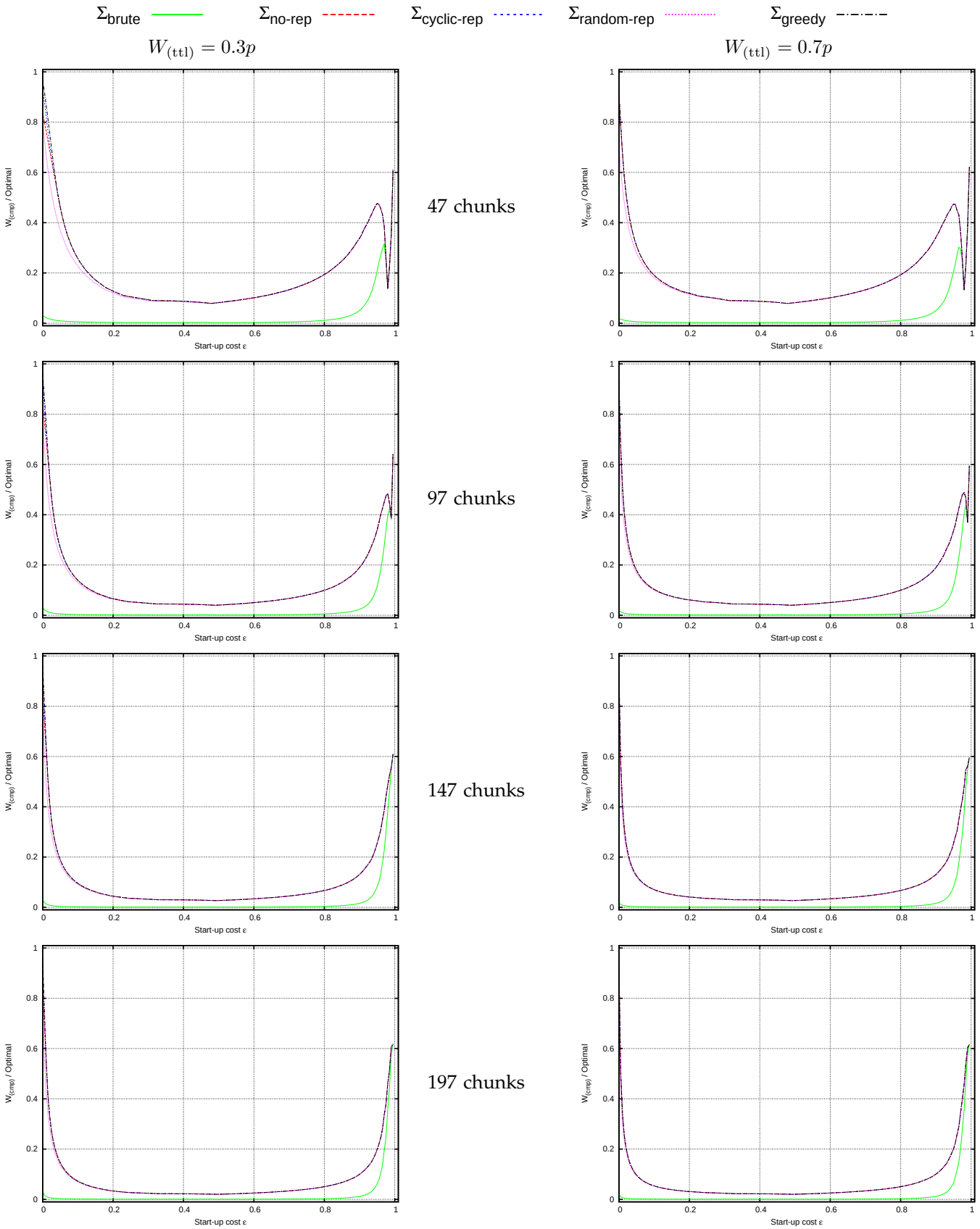


Fig. 33. Experiment (E4) with 100 computers.

APPENDIX B
EXPERIMENTS WITH LINEAR RISK FUNCTIONS (ALL HEURISTICS)

On the following graphs, the performance of all the heuristics is displayed, including all our group heuristics.

B.1 Experiments E1

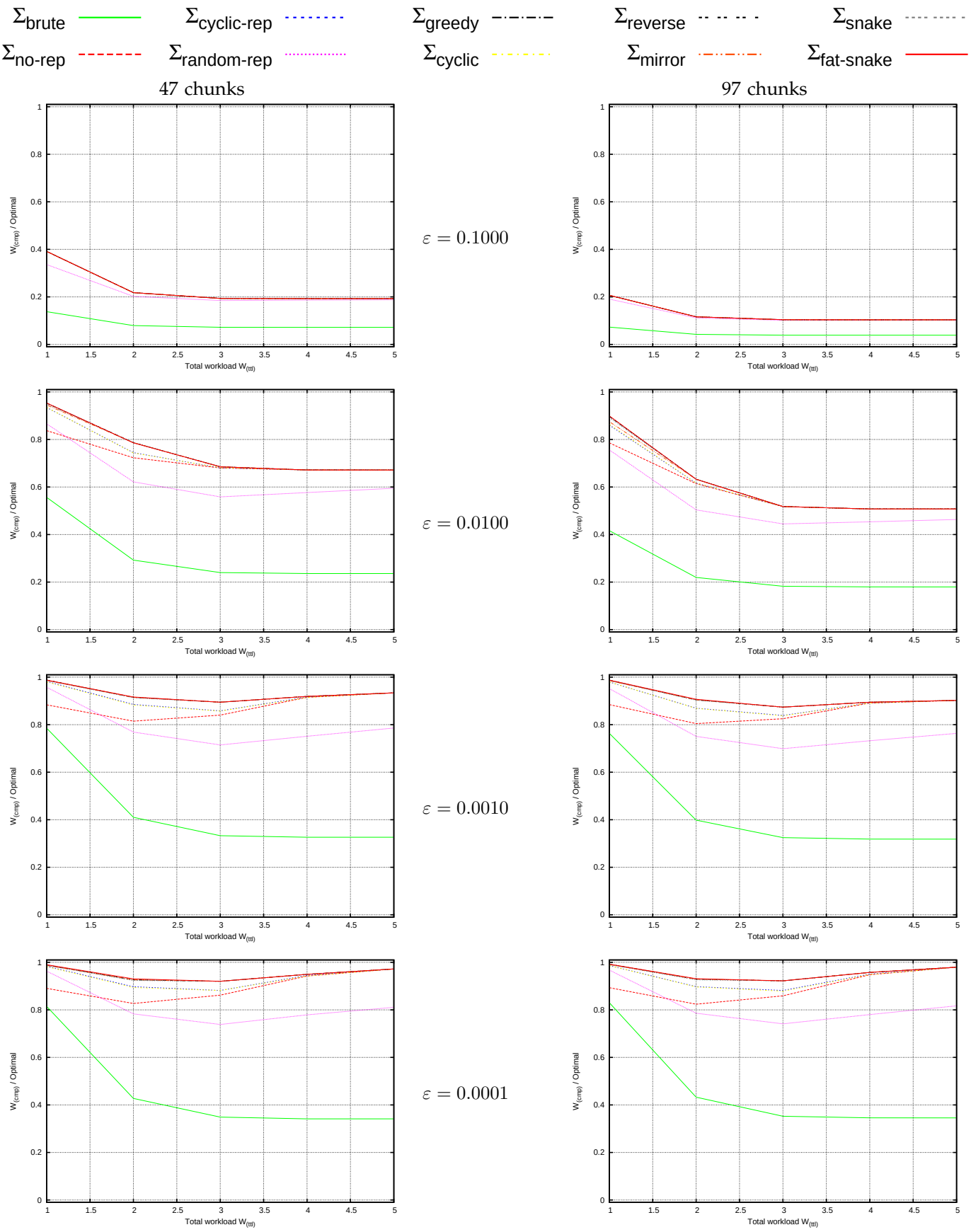


Fig. 34. Experiment (E1) using 5 computers.

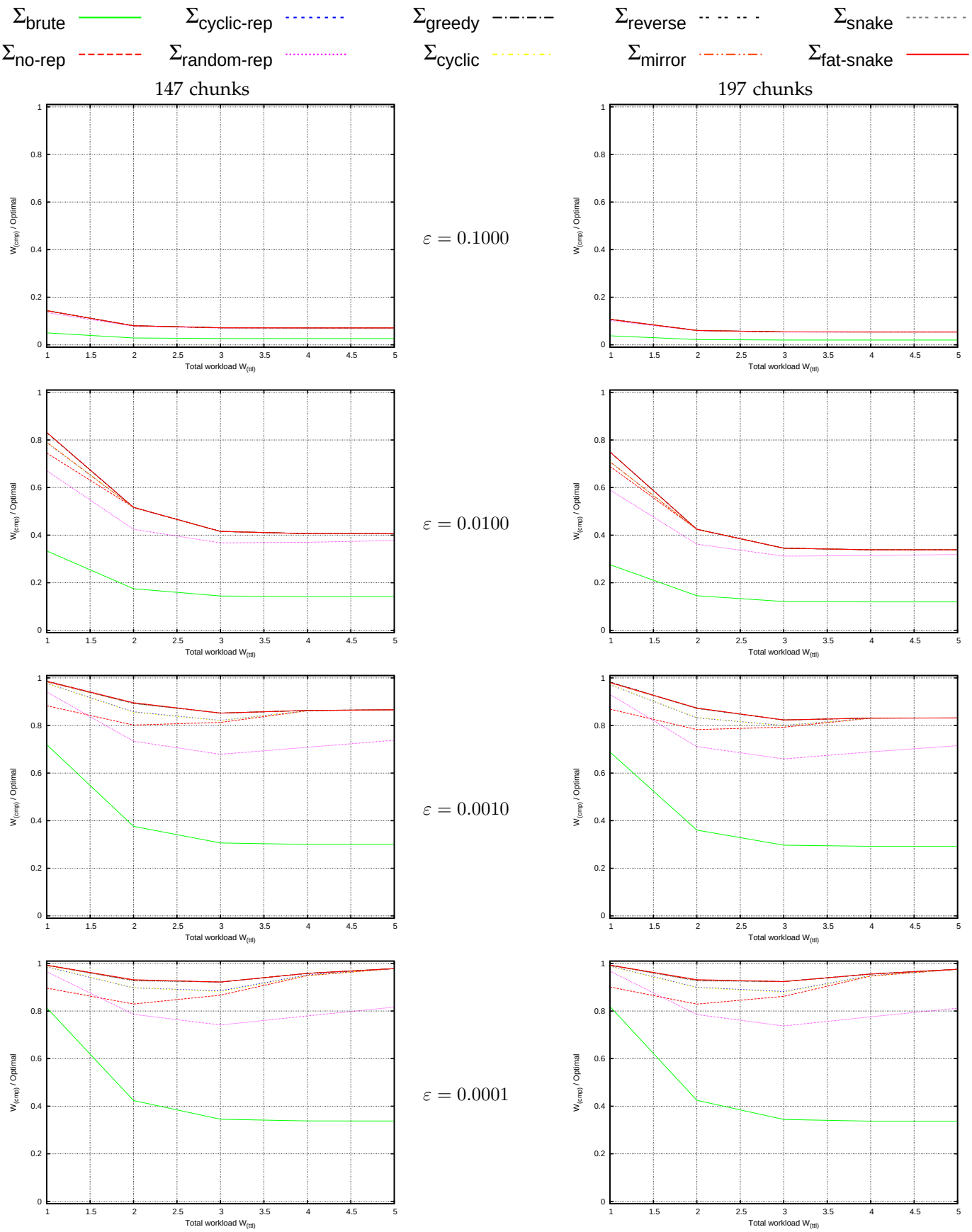


Fig. 35. Experiment (E1) using 5 computers (continued).

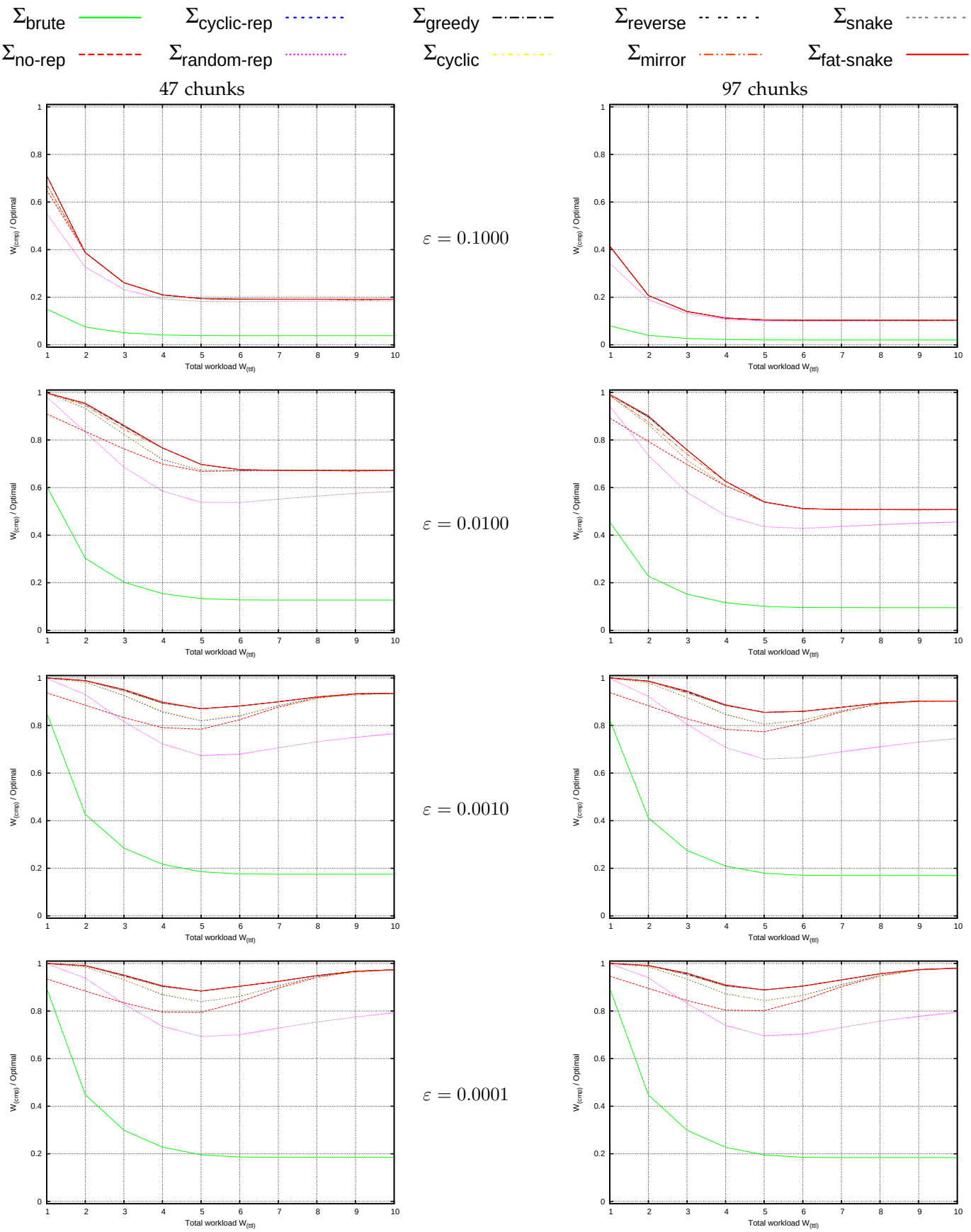


Fig. 36. Experiment (E1) using 10 computers.

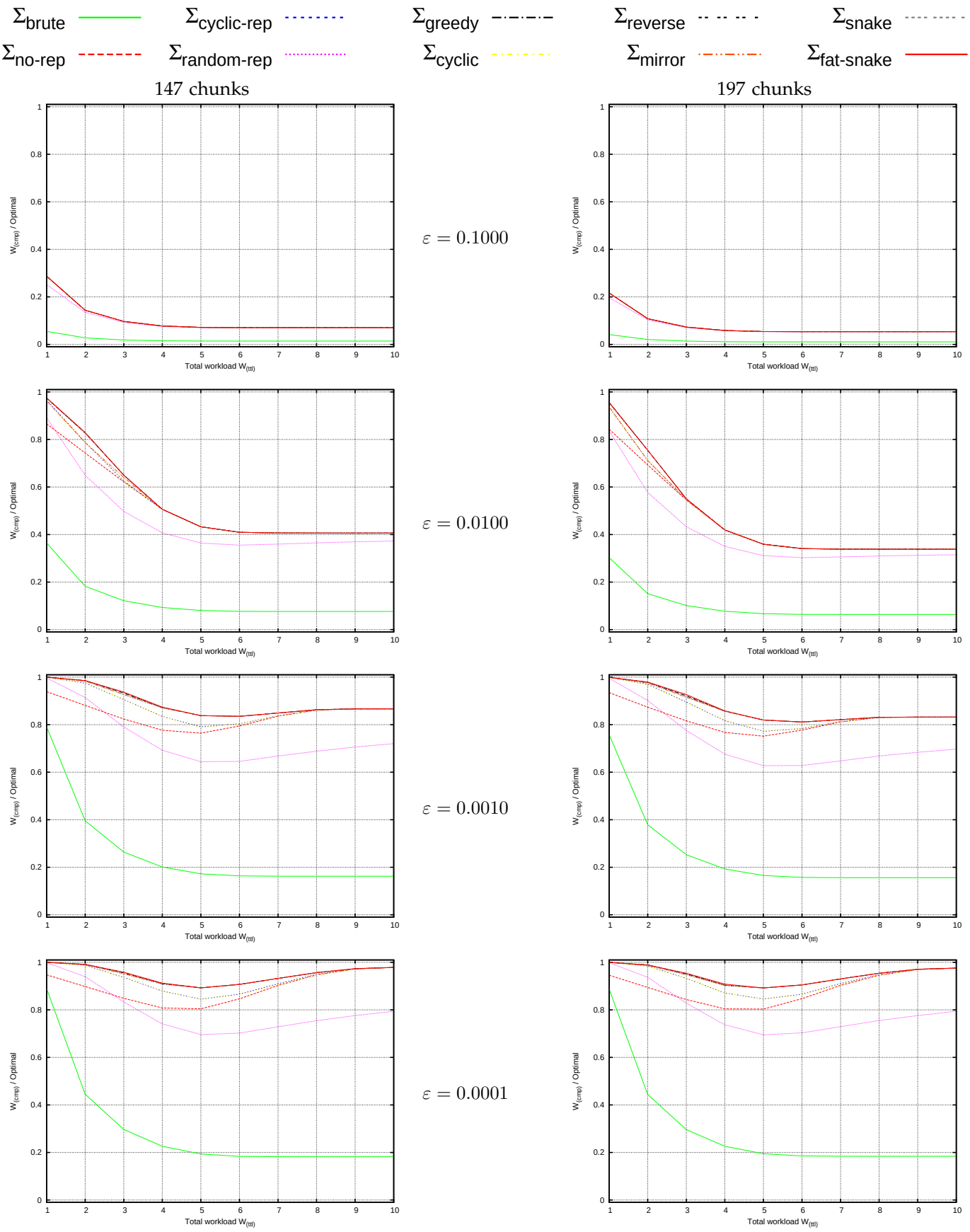


Fig. 37. Experiment (E1) using 10 computers (continued).

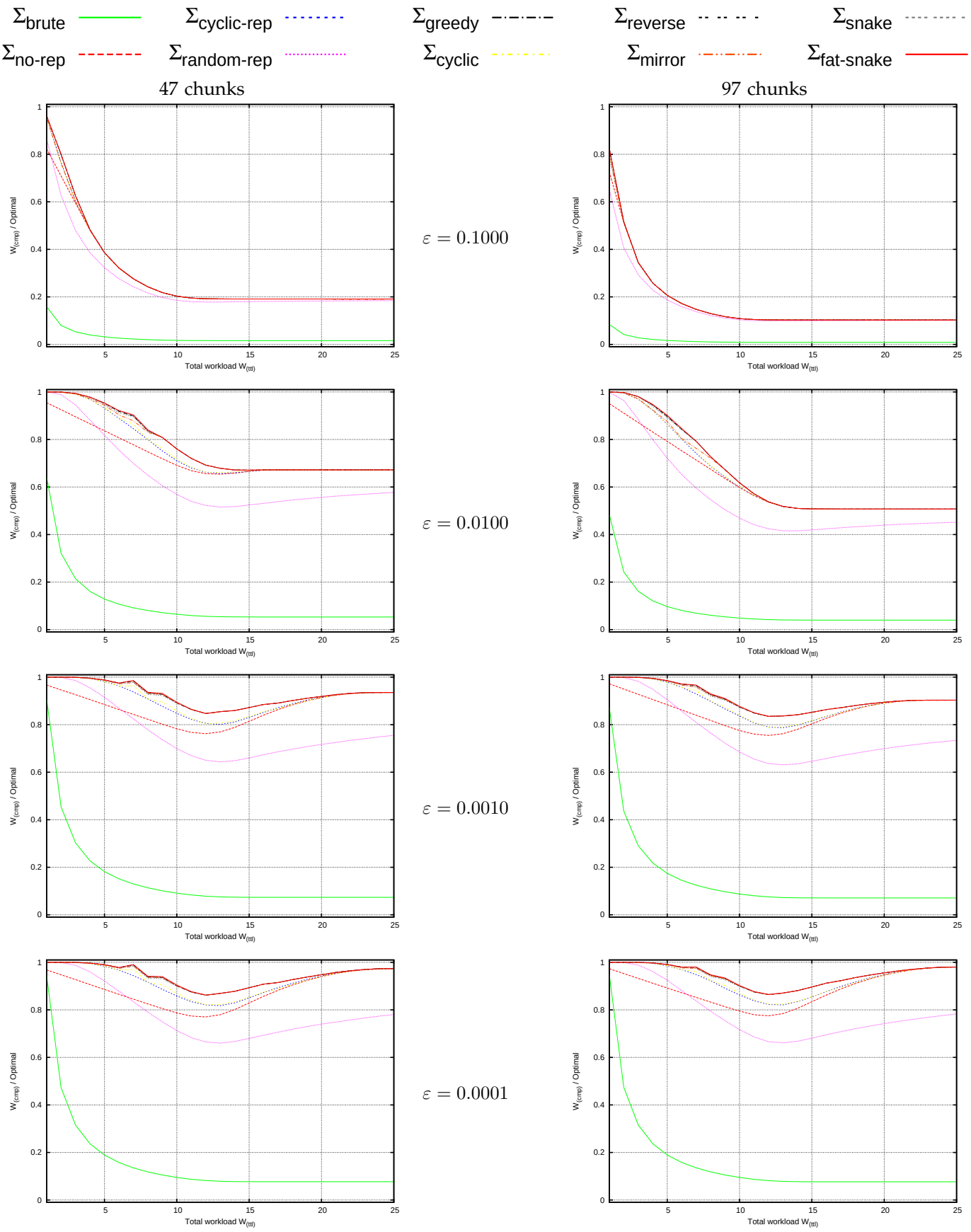


Fig. 38. Experiment (E1) using 25 computers.

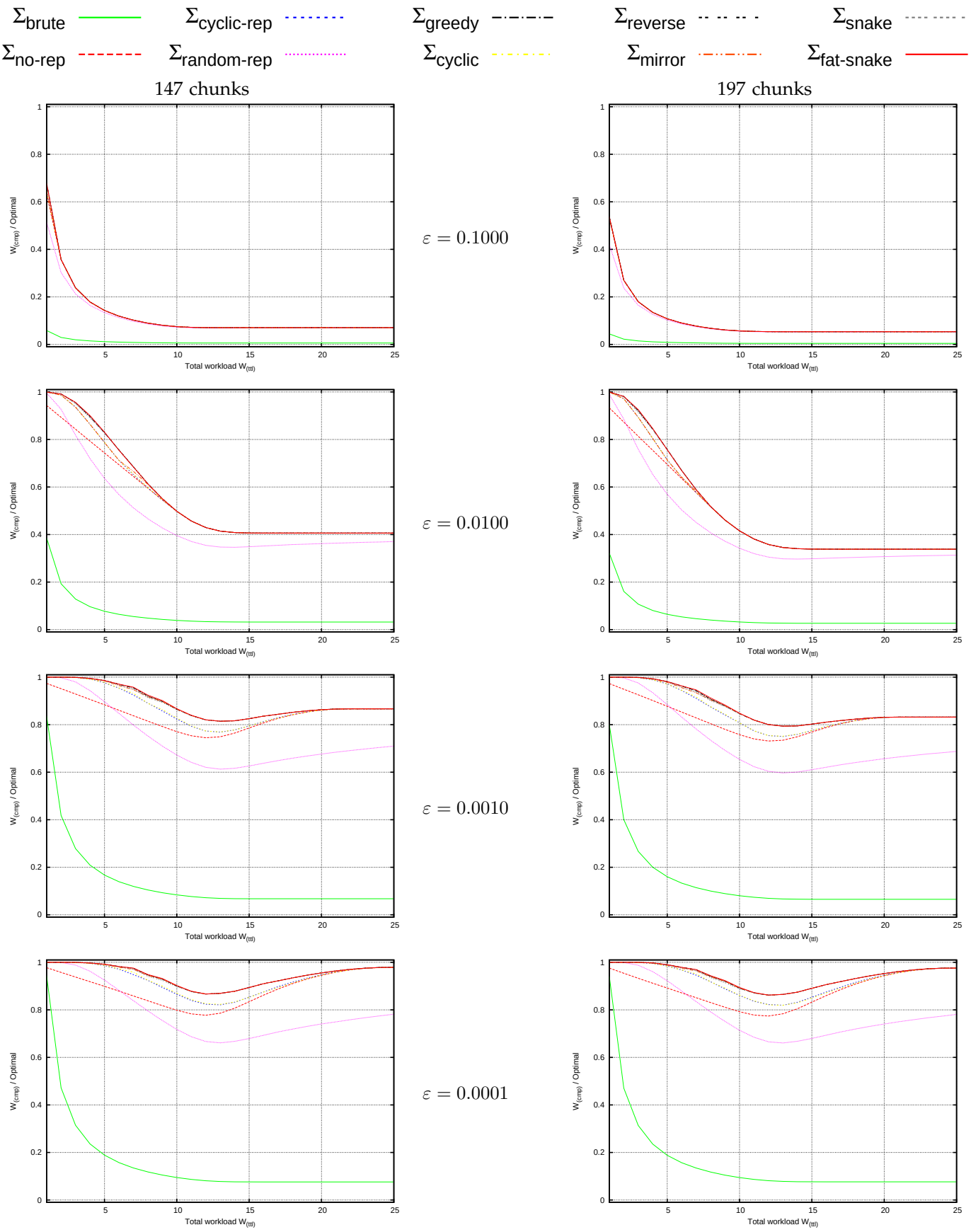


Fig. 39. Experiment (E1) using 25 computers (continued).

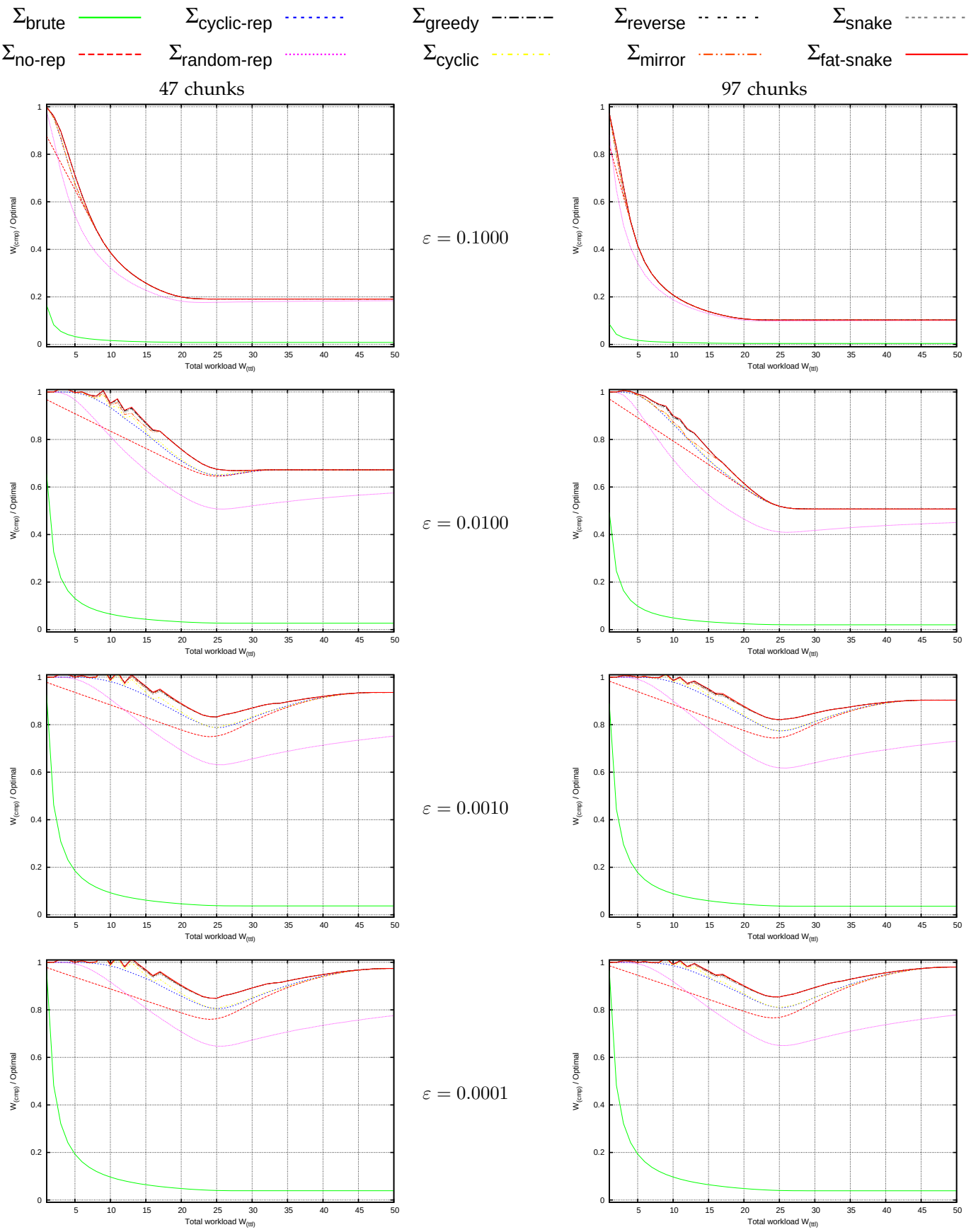


Fig. 40. Experiment (E1) using 50 computers.

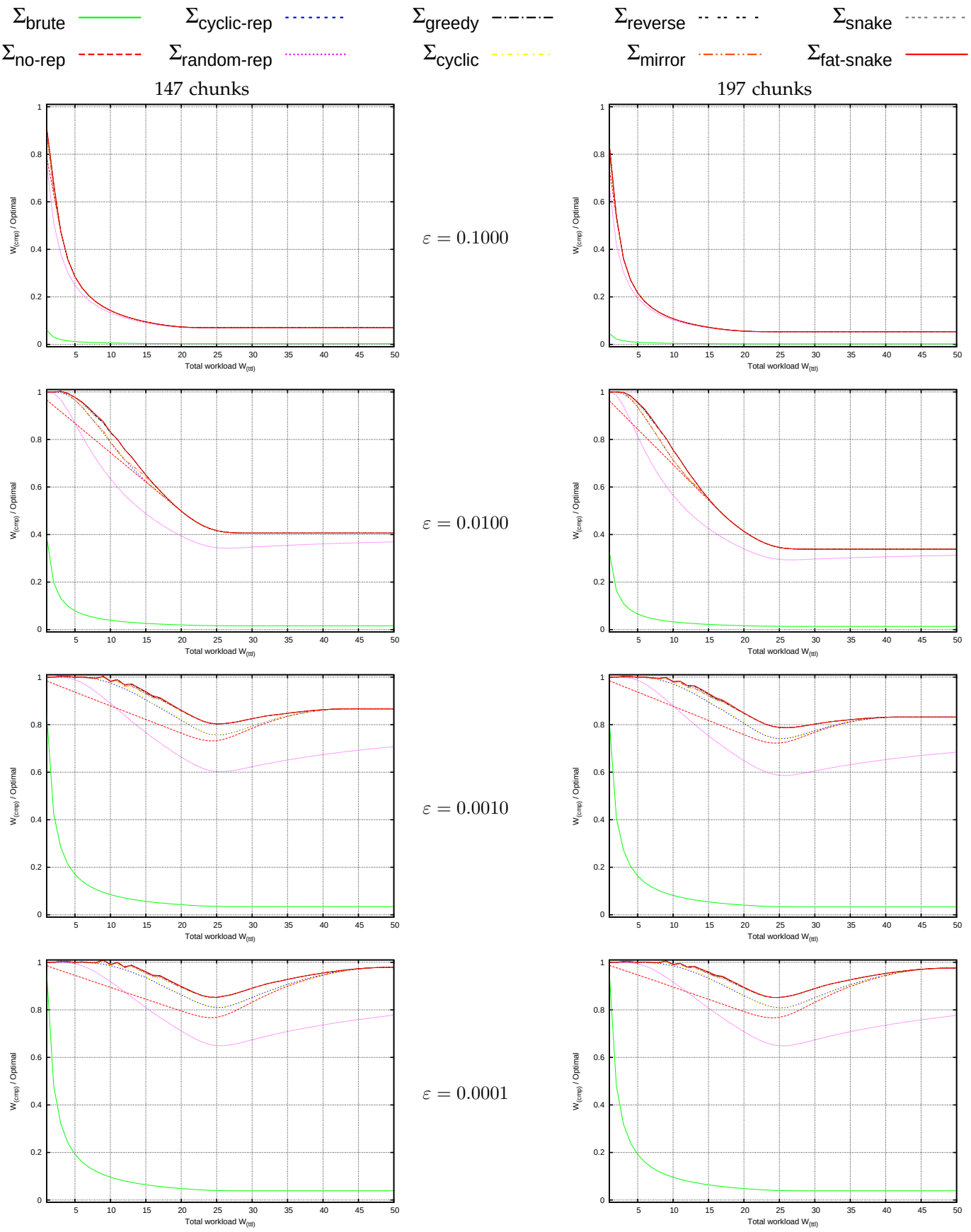


Fig. 41. Experiment (E1) using 50 computers (continued).

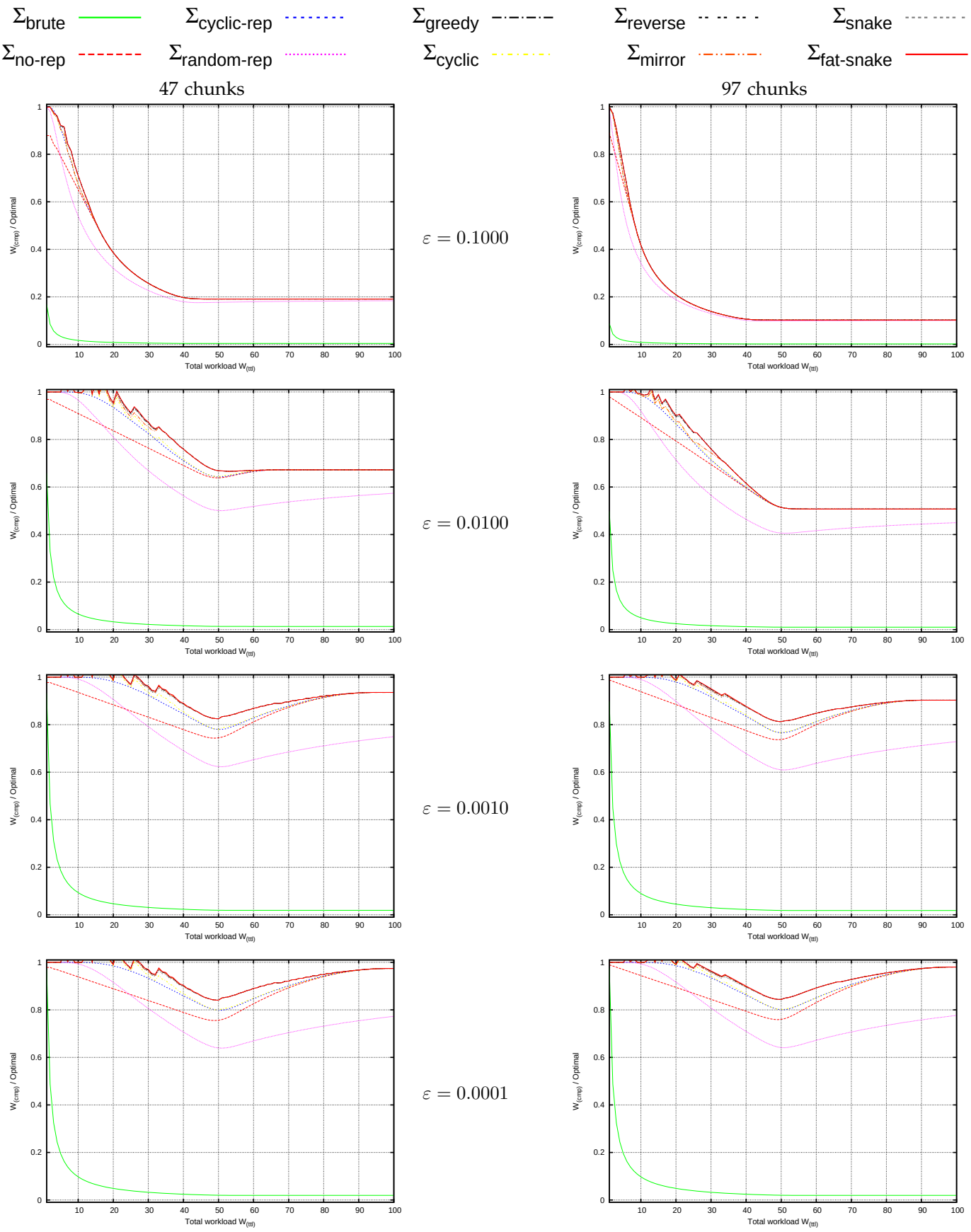


Fig. 42. Experiment (E1) using 100 computers.

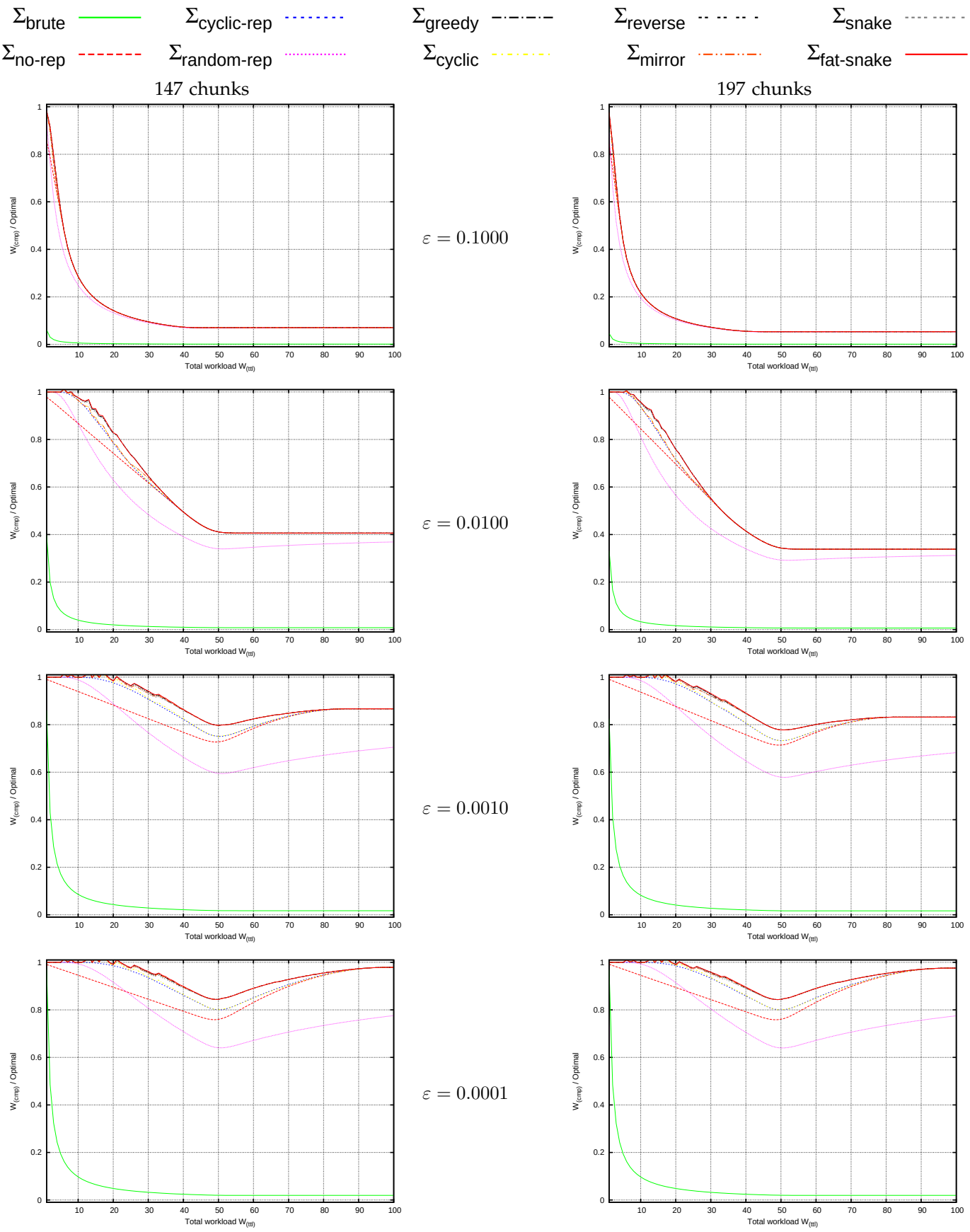


Fig. 43. Experiment (E1) using 100 computers (continued).

B.2 Experiments E2

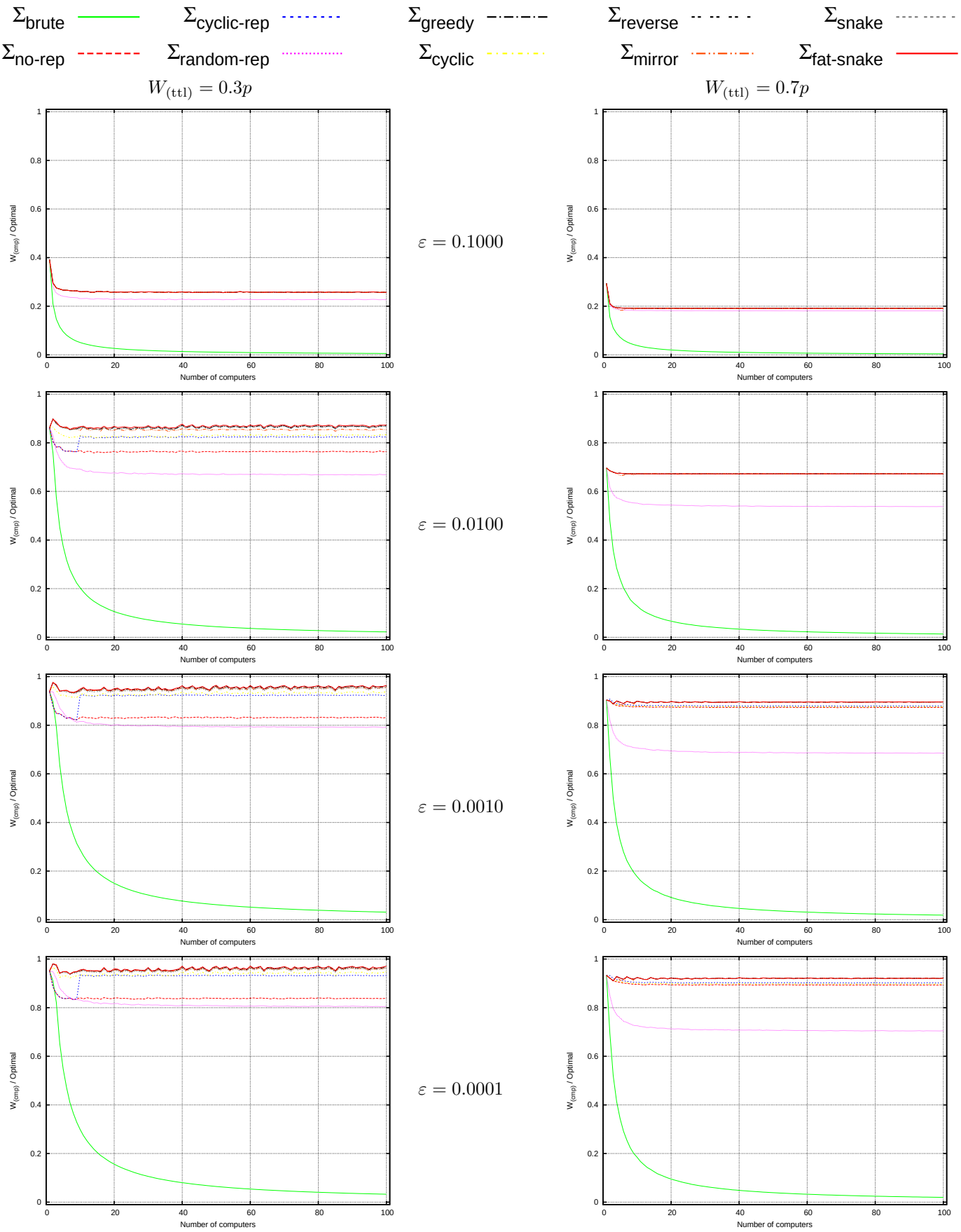


Fig. 44. Experiment (E2) with 47 chunks.

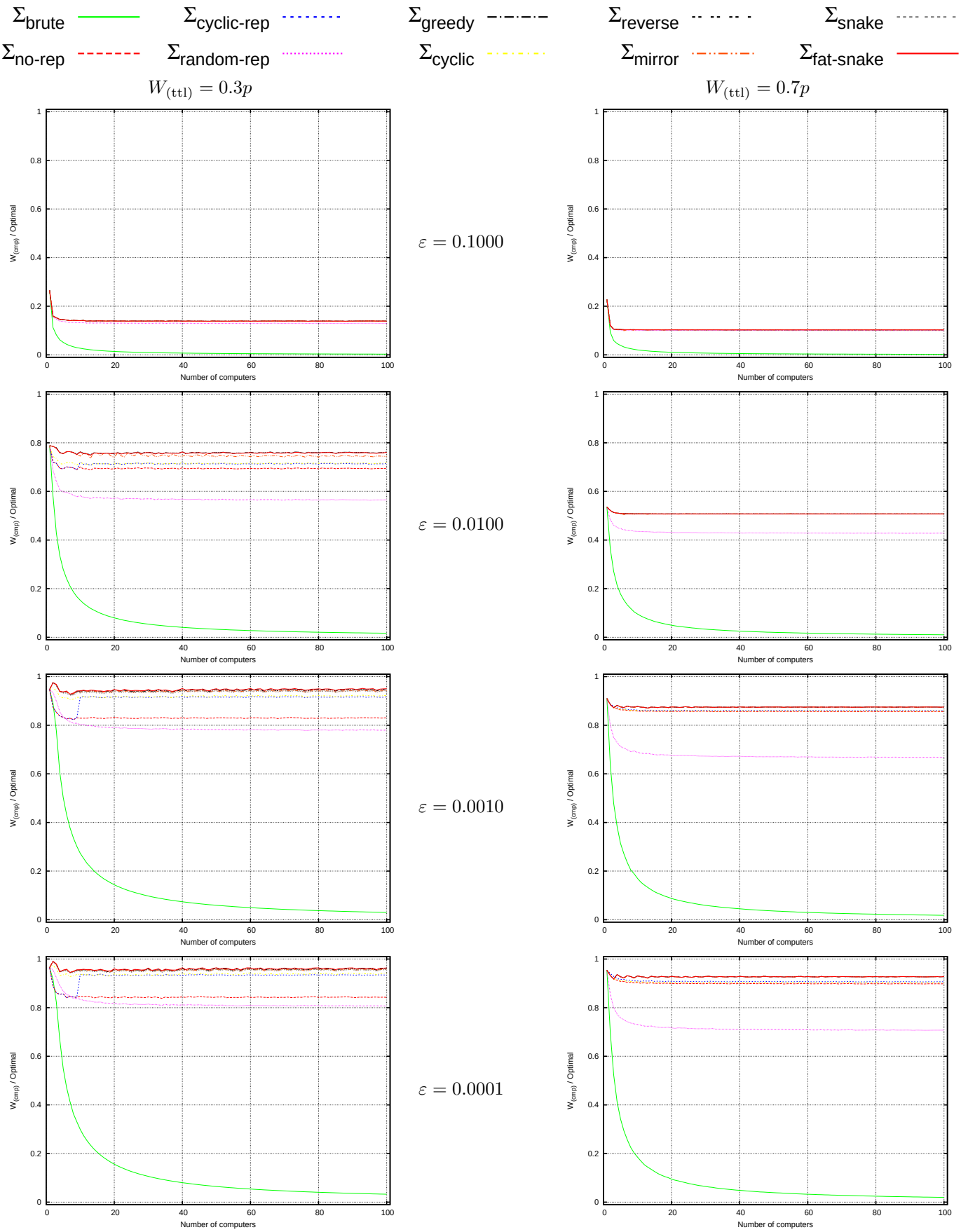


Fig. 45. Experiment (E2) with 97 chunks.

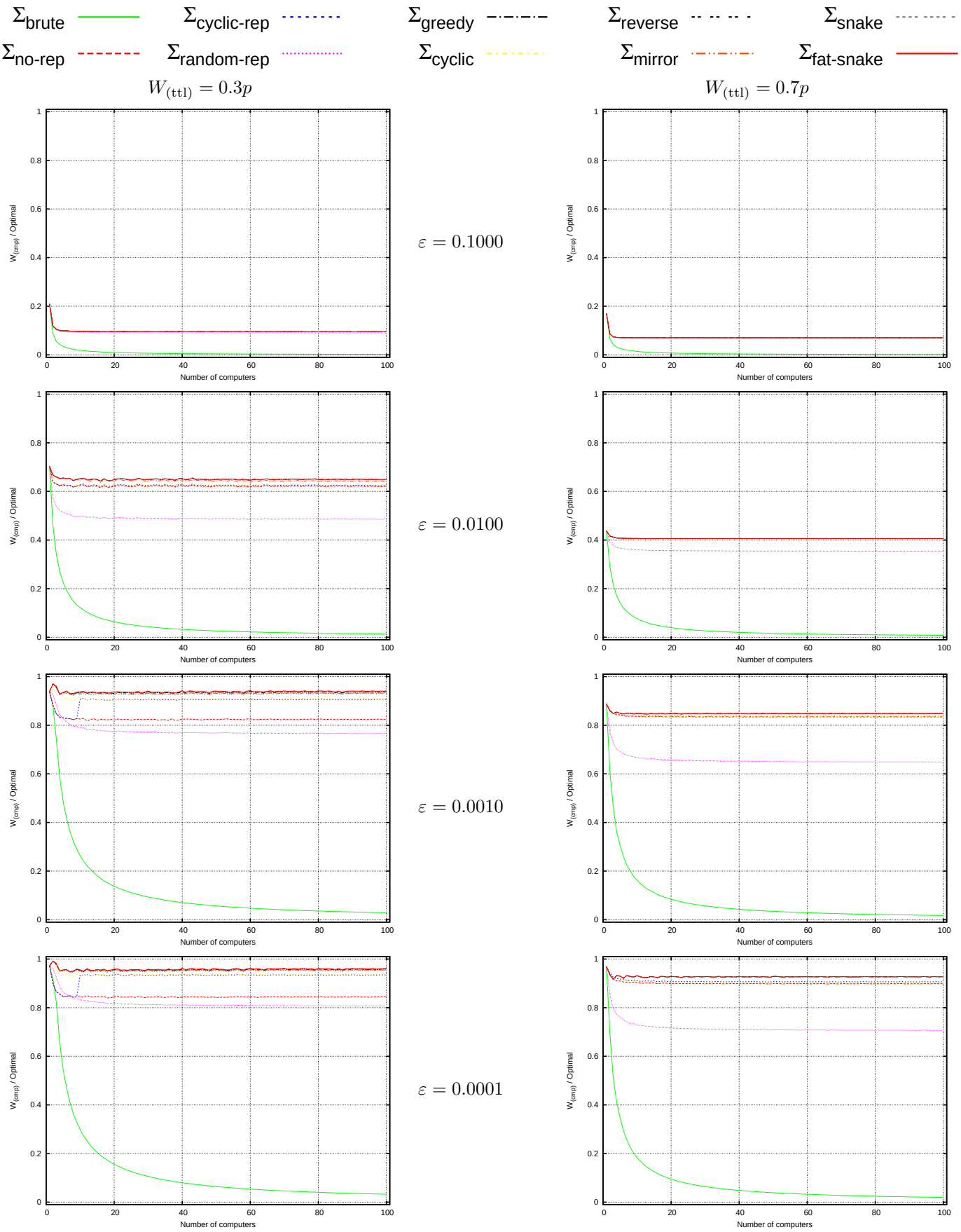


Fig. 46. Experiment (E2) with 147 chunks.

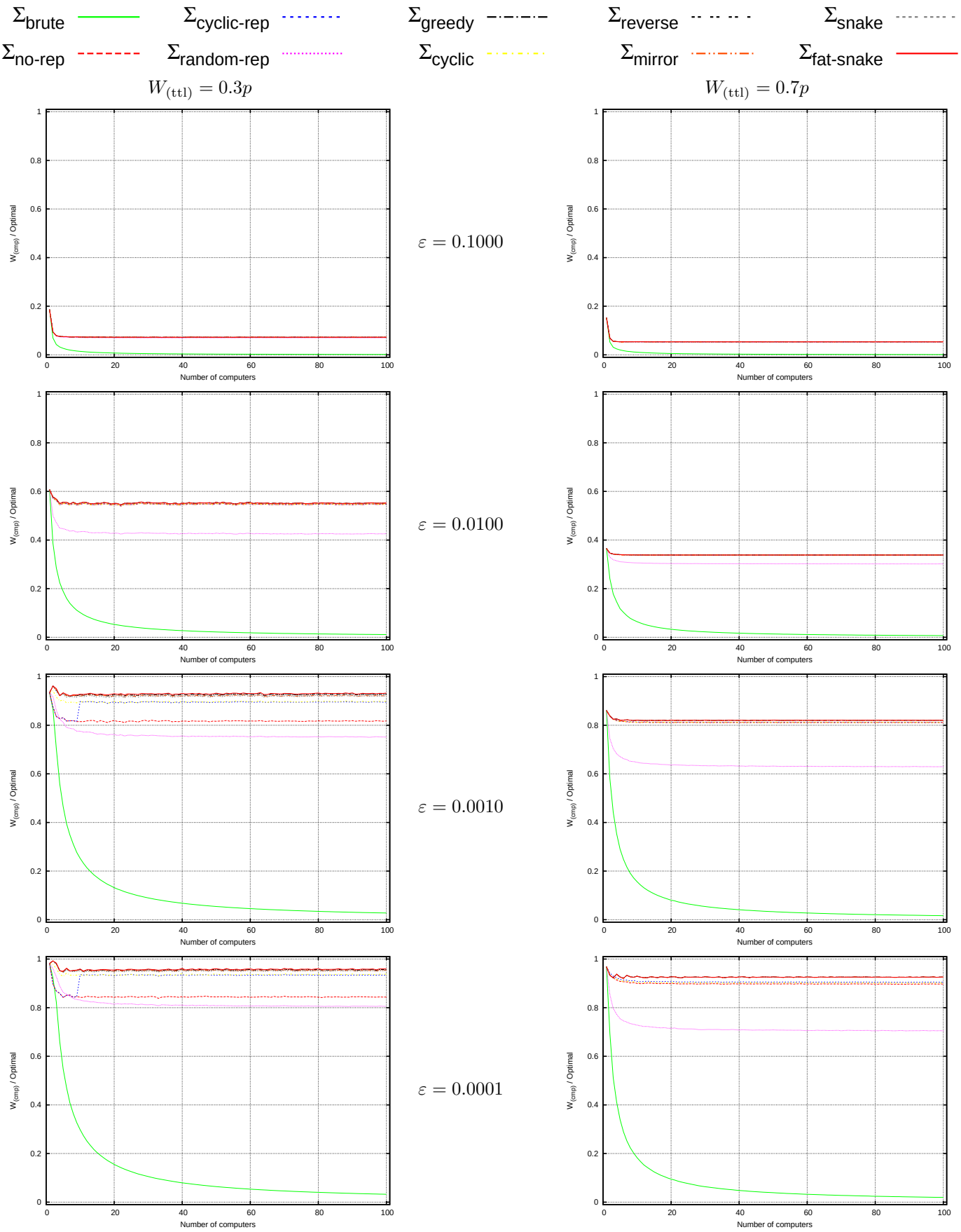


Fig. 47. Experiment (E2) with 197 chunks.

B.3 Experiments E3

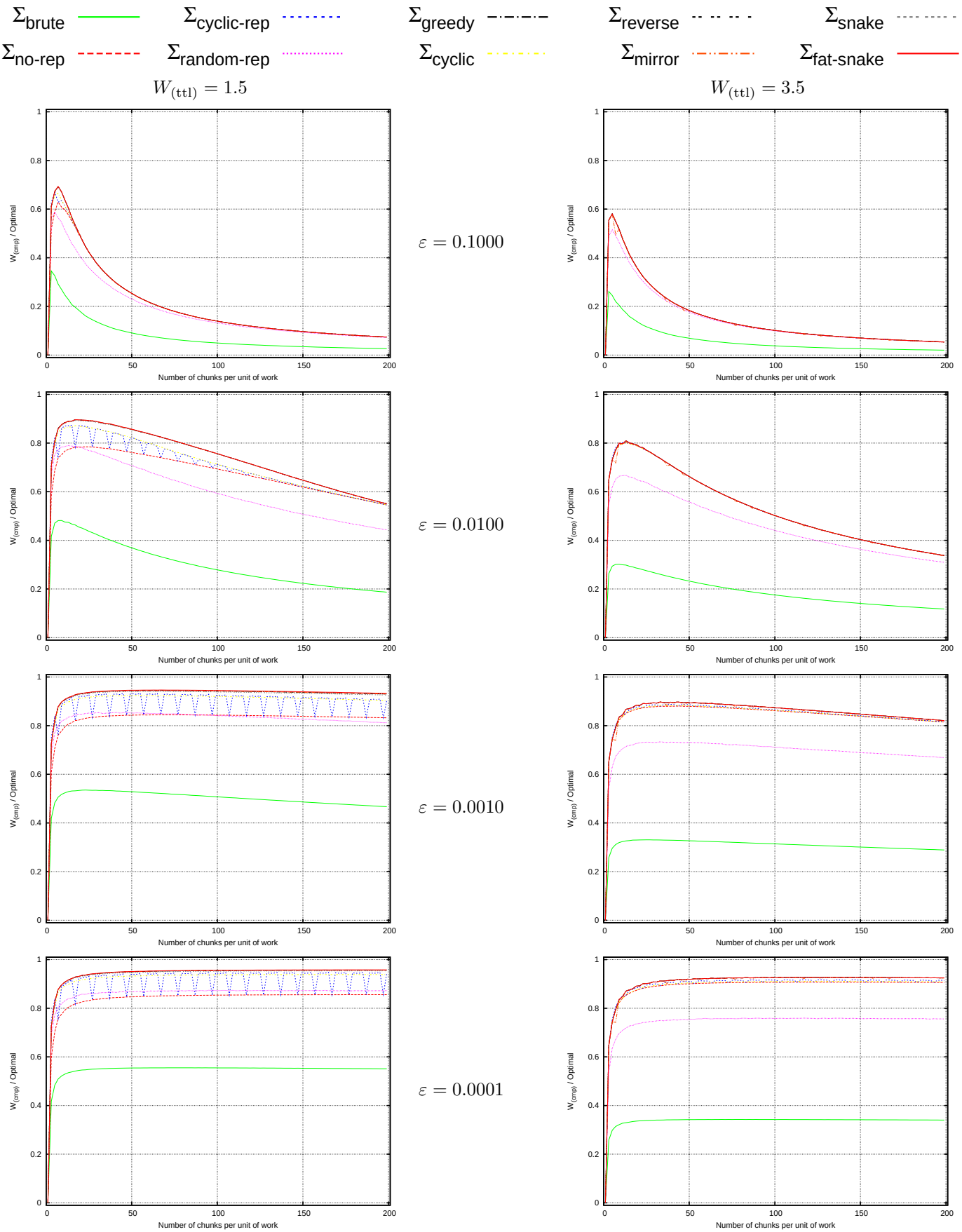


Fig. 48. Experiment (E3) using 5 computers.

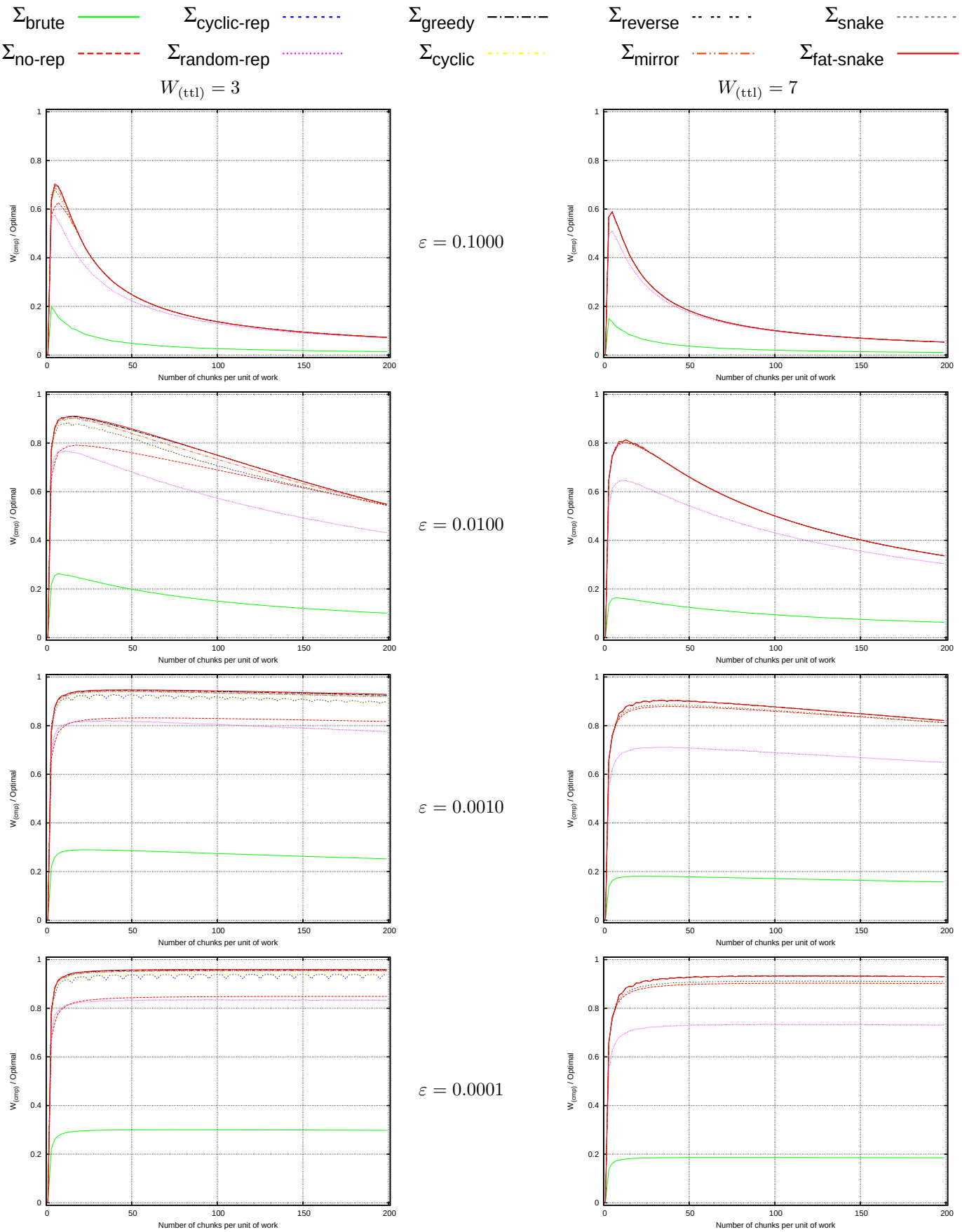


Fig. 49. Experiment (E3) using 10 computers.

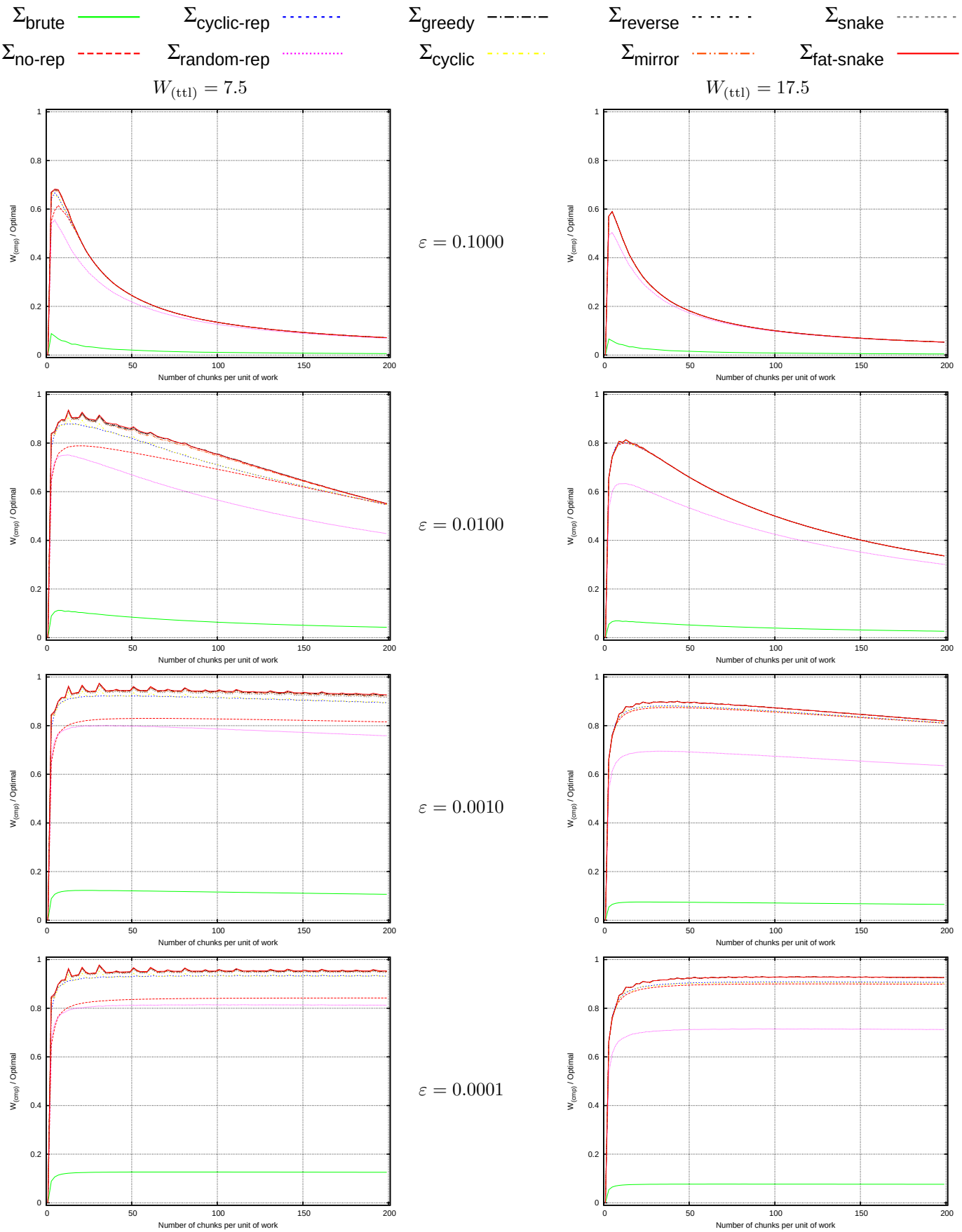


Fig. 50. Experiment (E3) using 25 computers.

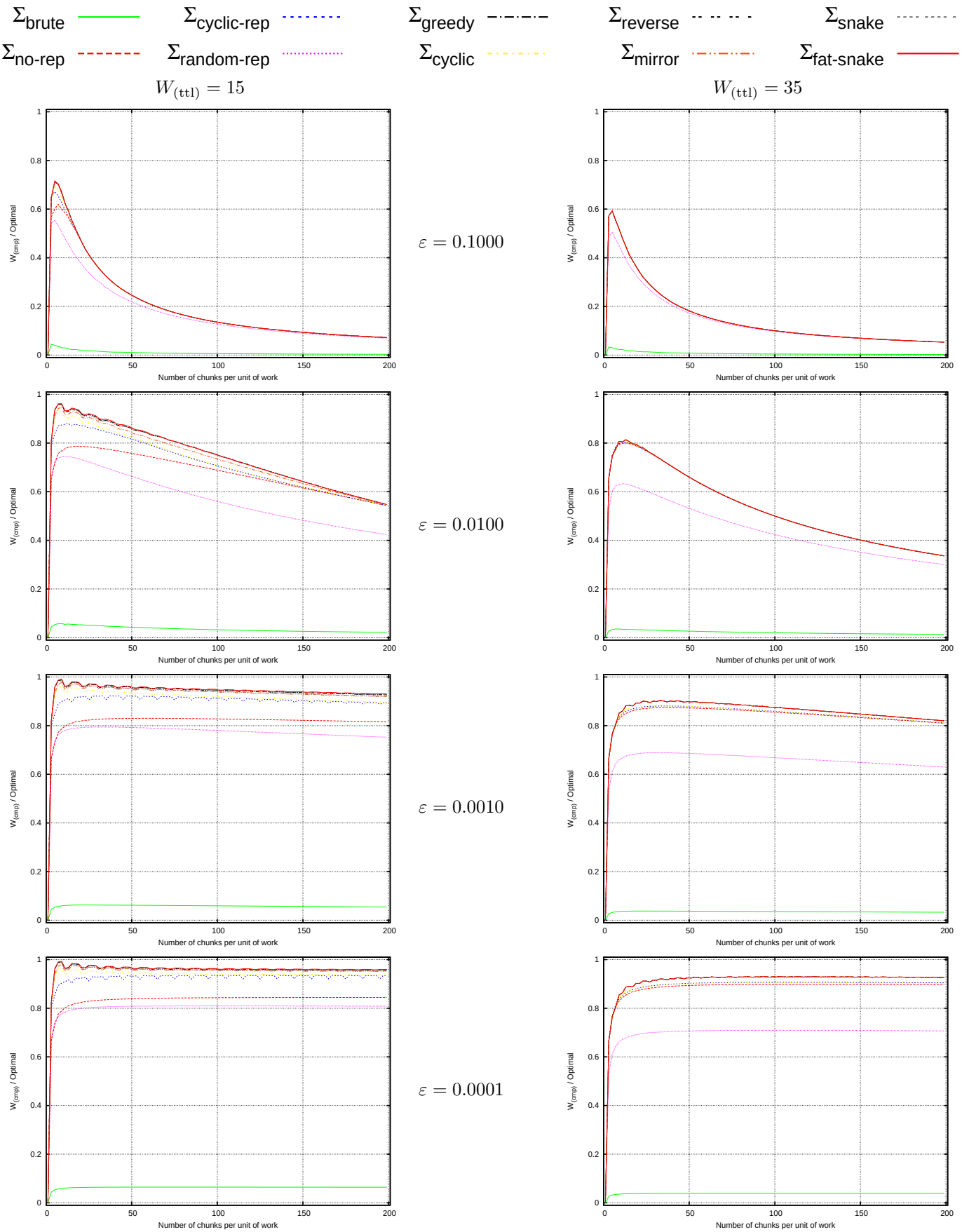


Fig. 51. Experiment (E3) using 50 computers.

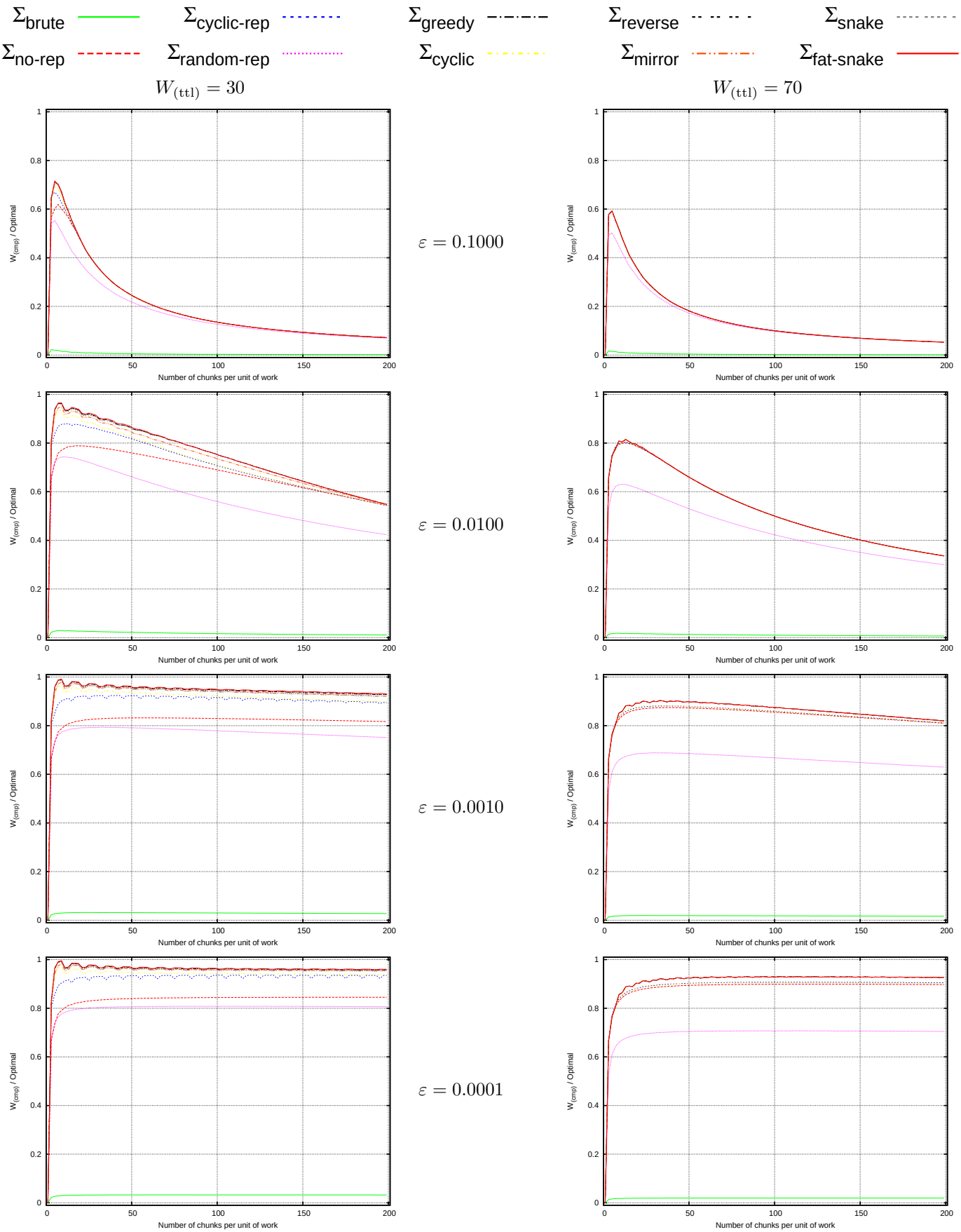


Fig. 52. Experiment (E3) using 100 computers.

B.4 Experiments E4

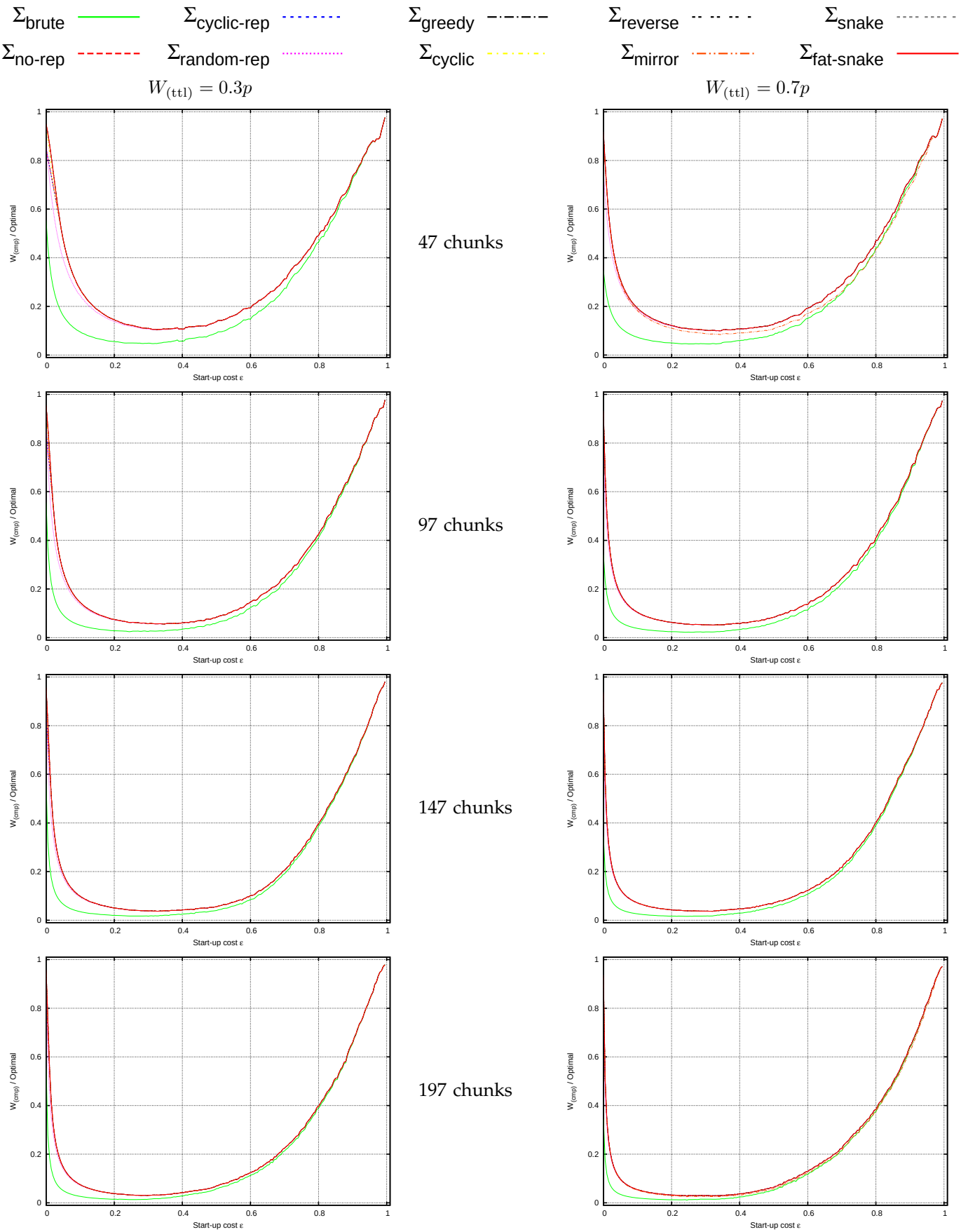


Fig. 53. Experiment (E4) with 5 computers.

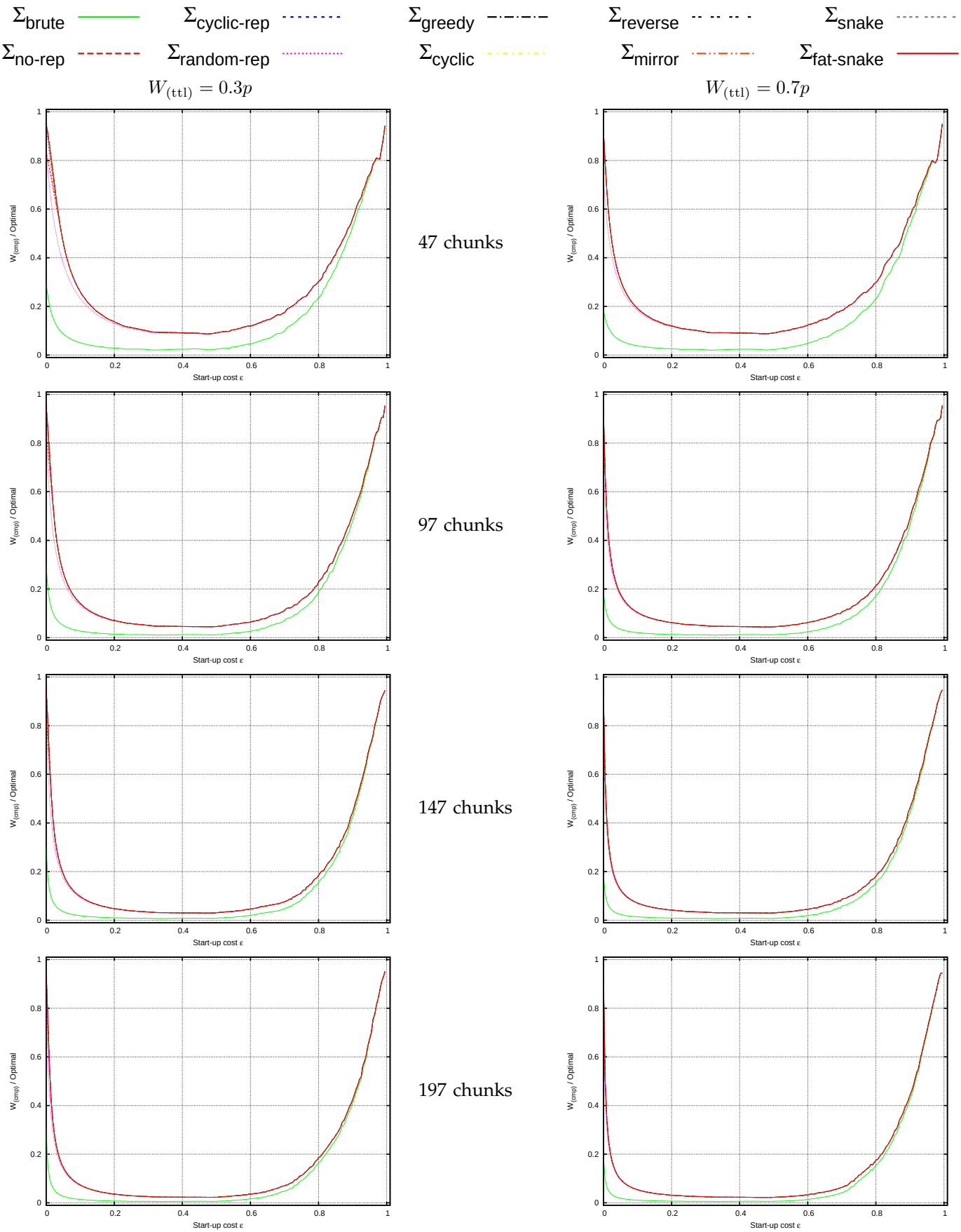


Fig. 54. Experiment (E4) with 10 computers.

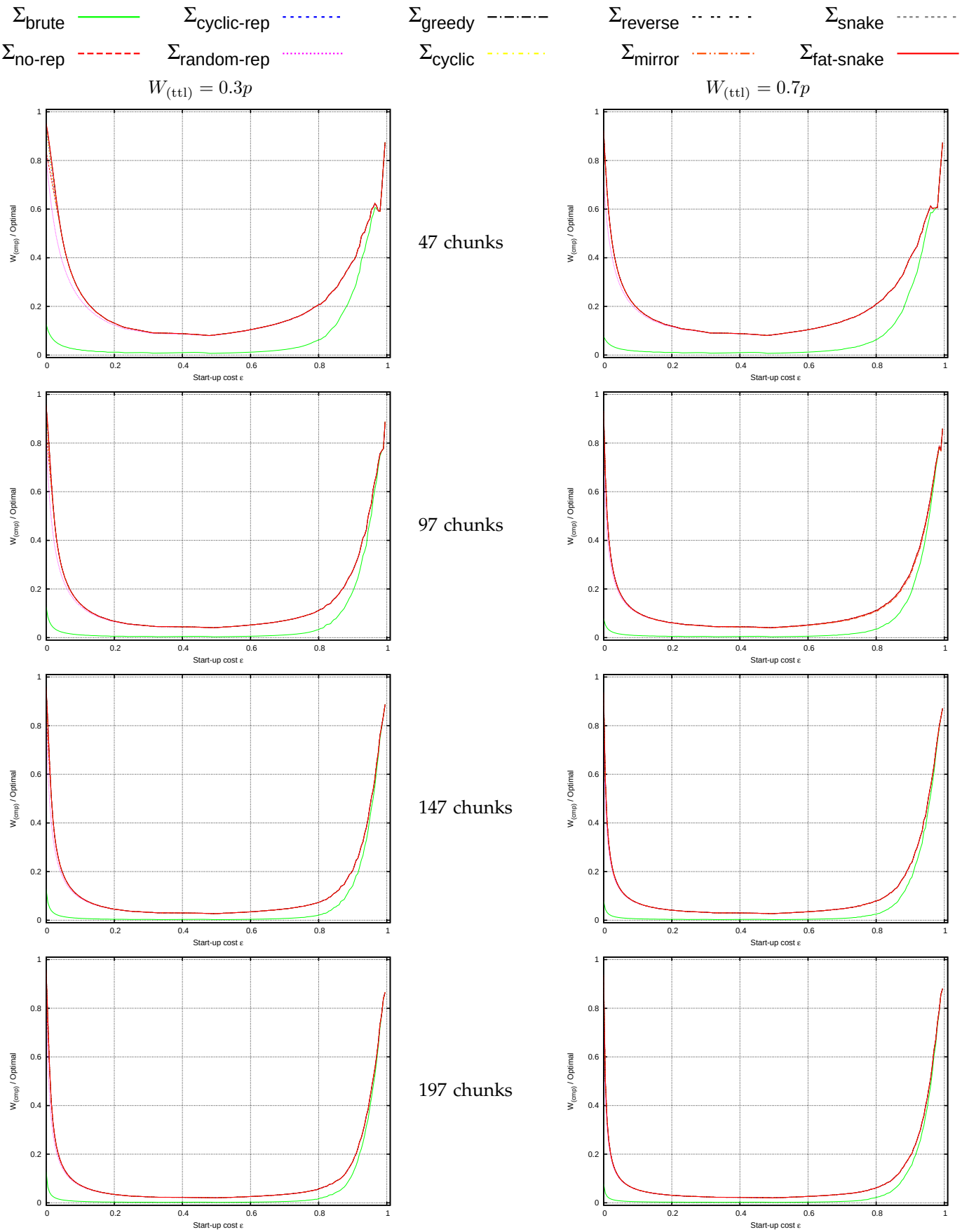


Fig. 55. Experiment (E4) with 25 computers.

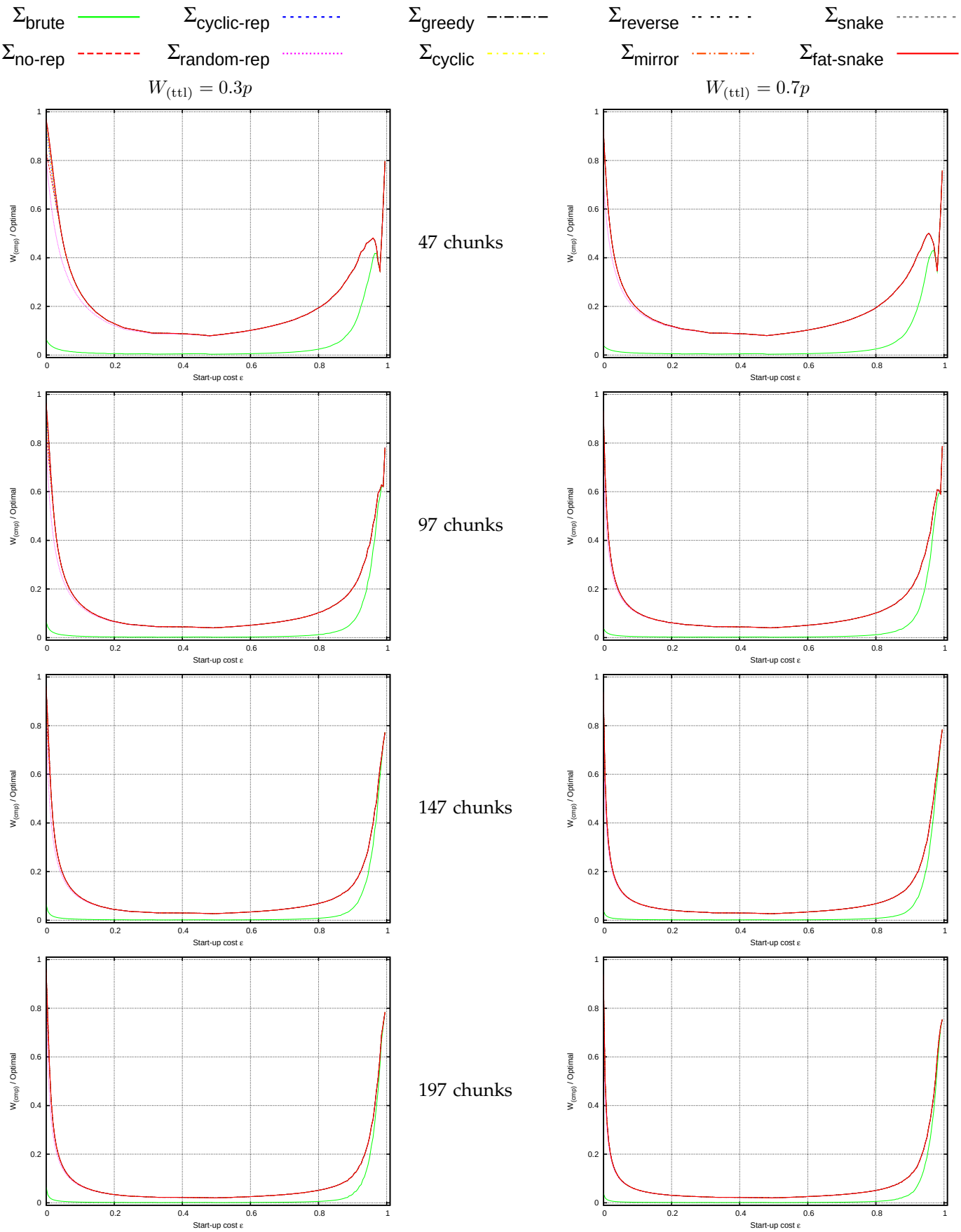


Fig. 56. Experiment (E4) with 50 computers.

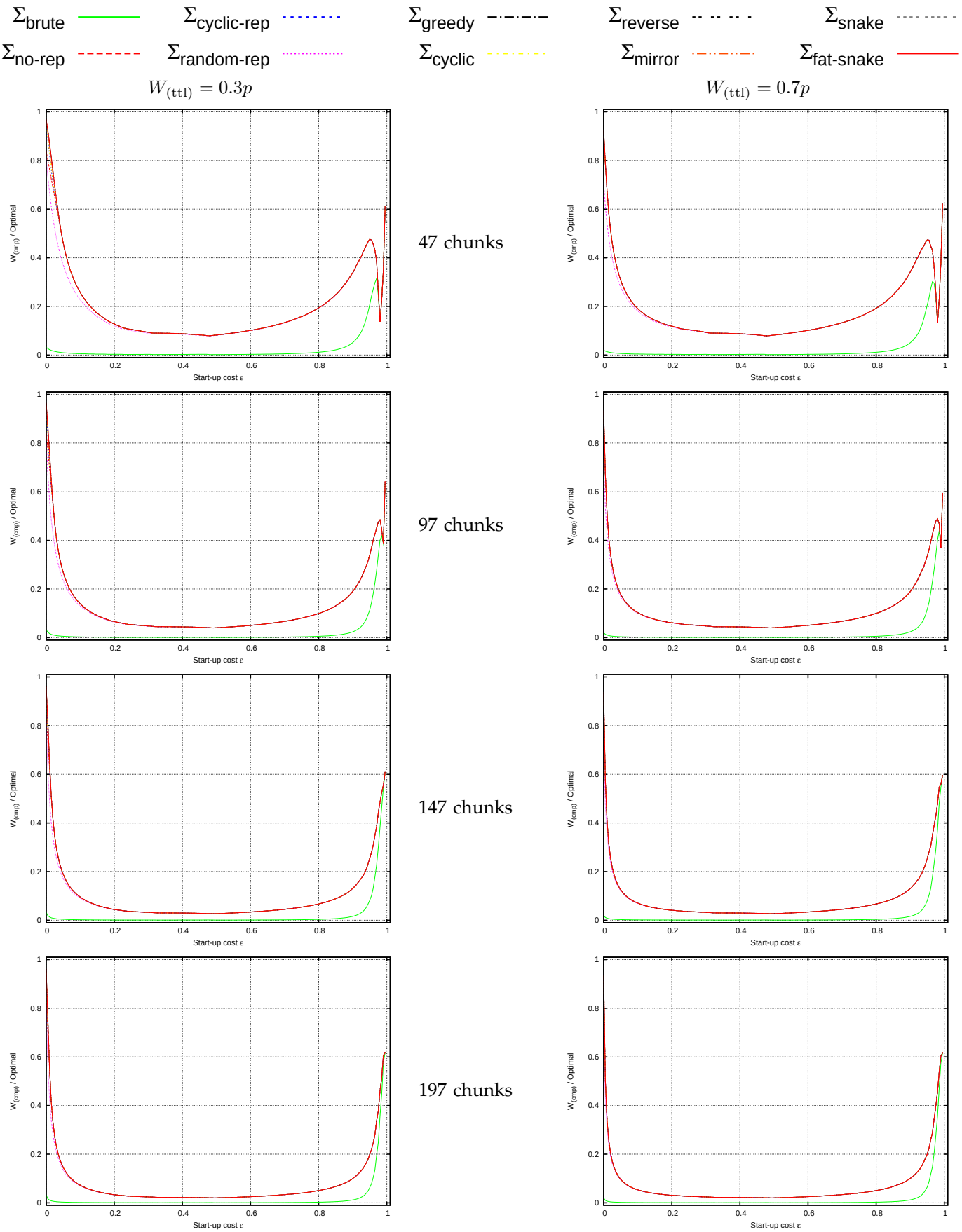
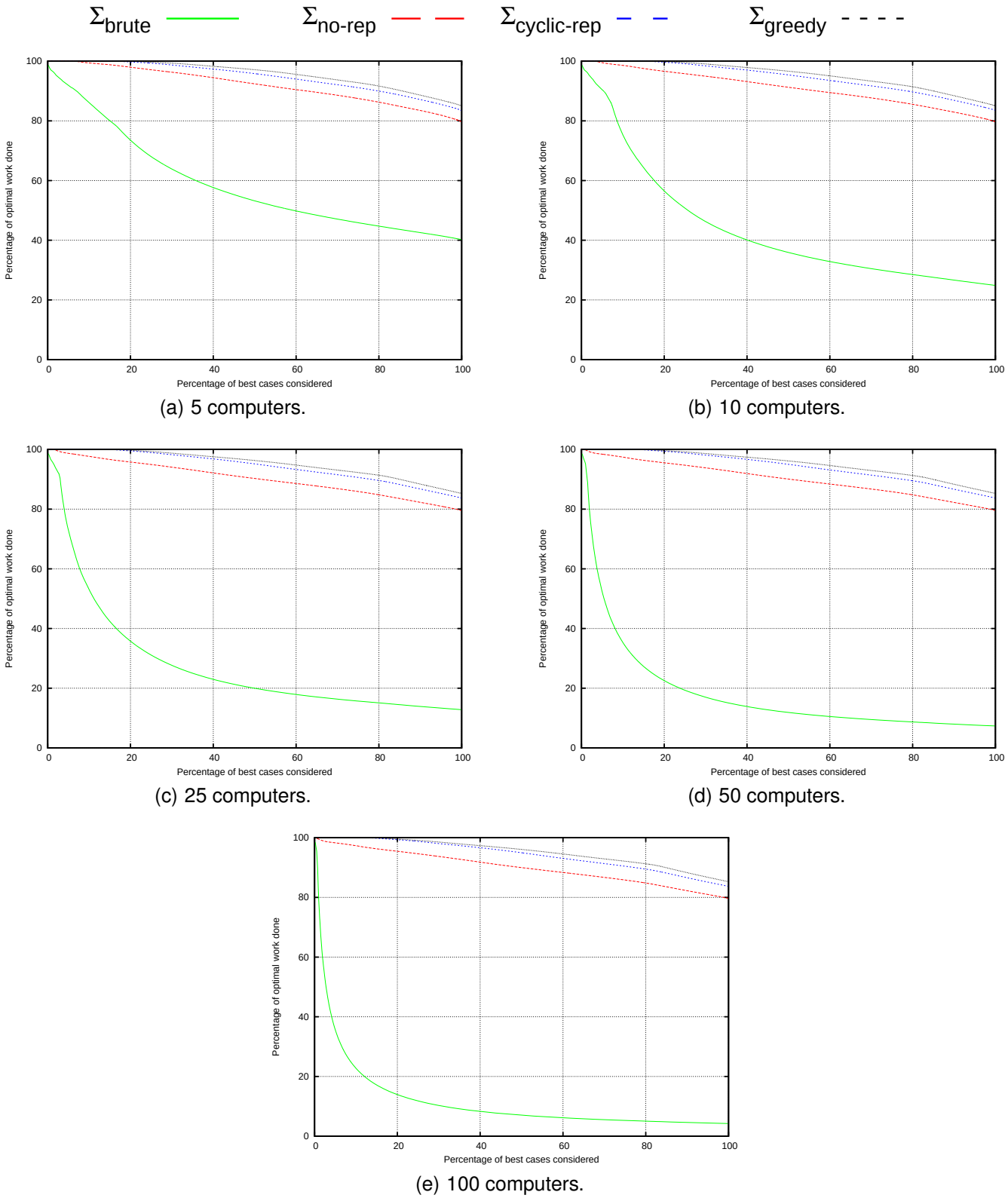


Fig. 57. Experiment (E4) with 100 computers.

B.5 Experiments E5



(a) 5 computers.

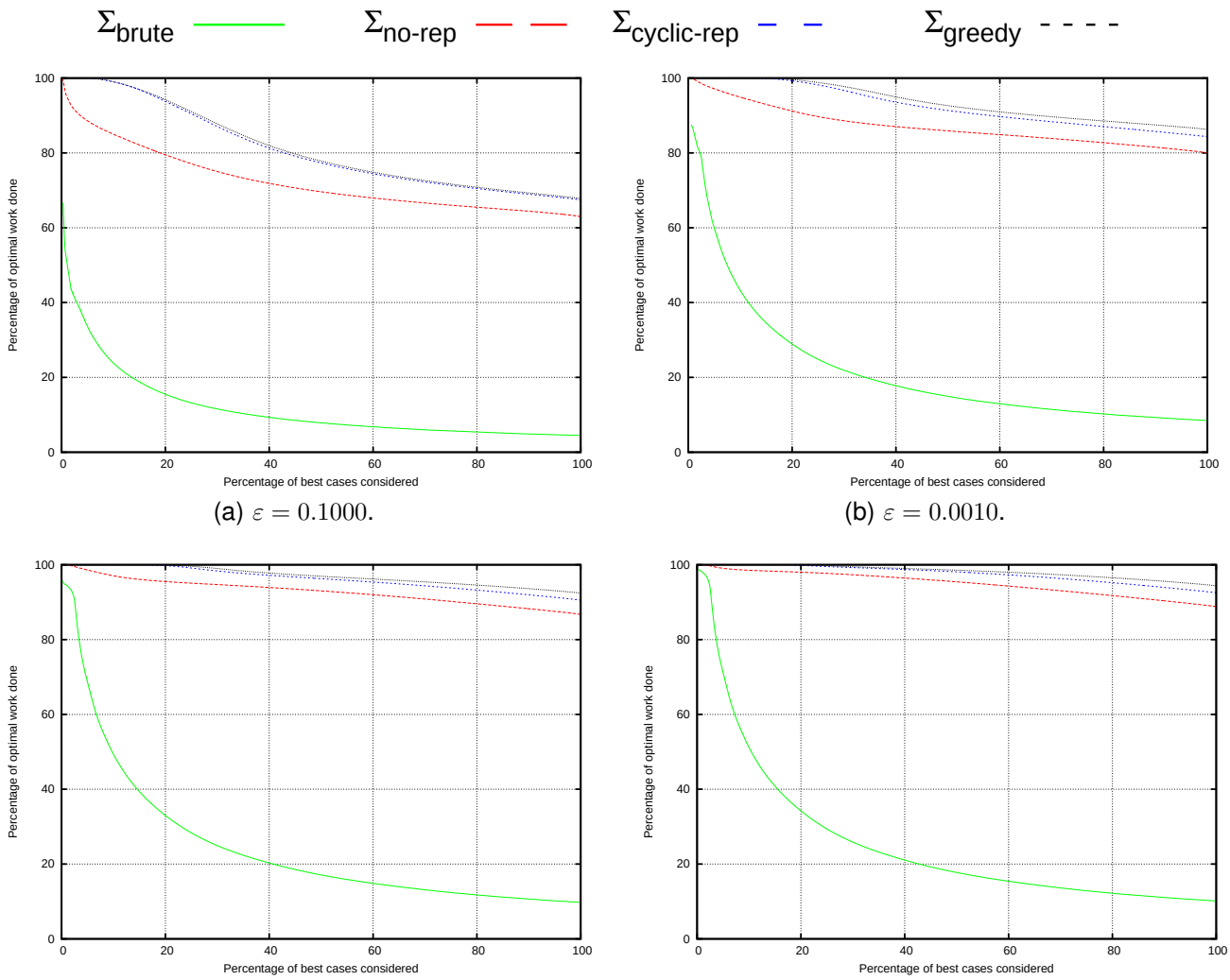
(b) 10 computers.

(c) 25 computers.

(d) 50 computers.

(e) 100 computers.

Fig. 58. Experiment (E5) using different numbers of computers.



(a) $\varepsilon = 0.1000$.

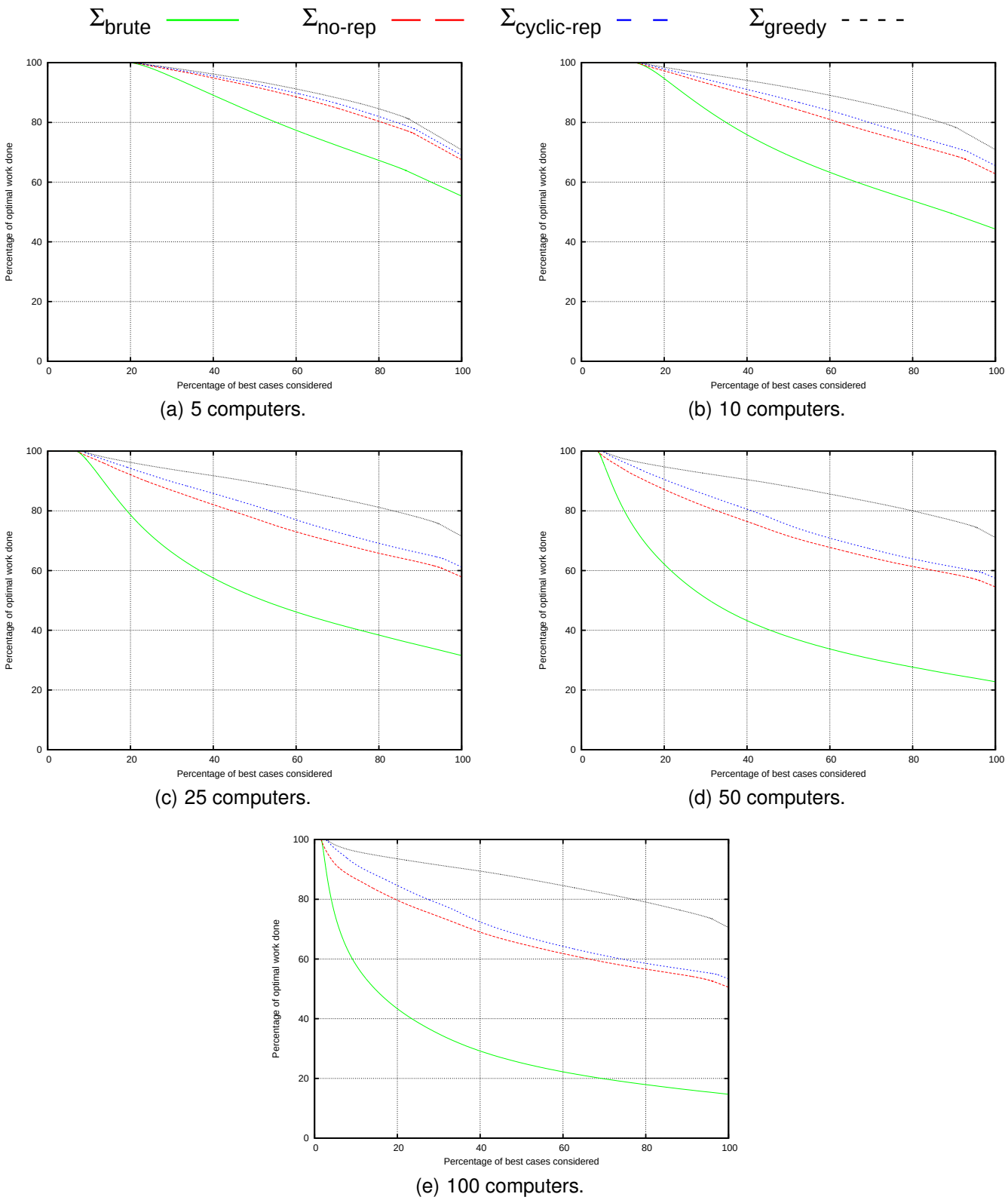
(b) $\varepsilon = 0.0010$.

(c) $\varepsilon = 0.0010$.

(d) $\varepsilon = 0.0001$.

Fig. 59. Experiment (E5) using different values of start-up cost.

APPENDIX C
EXPERIMENTS WITH GENERAL RISK FUNCTIONS



(a) 5 computers.

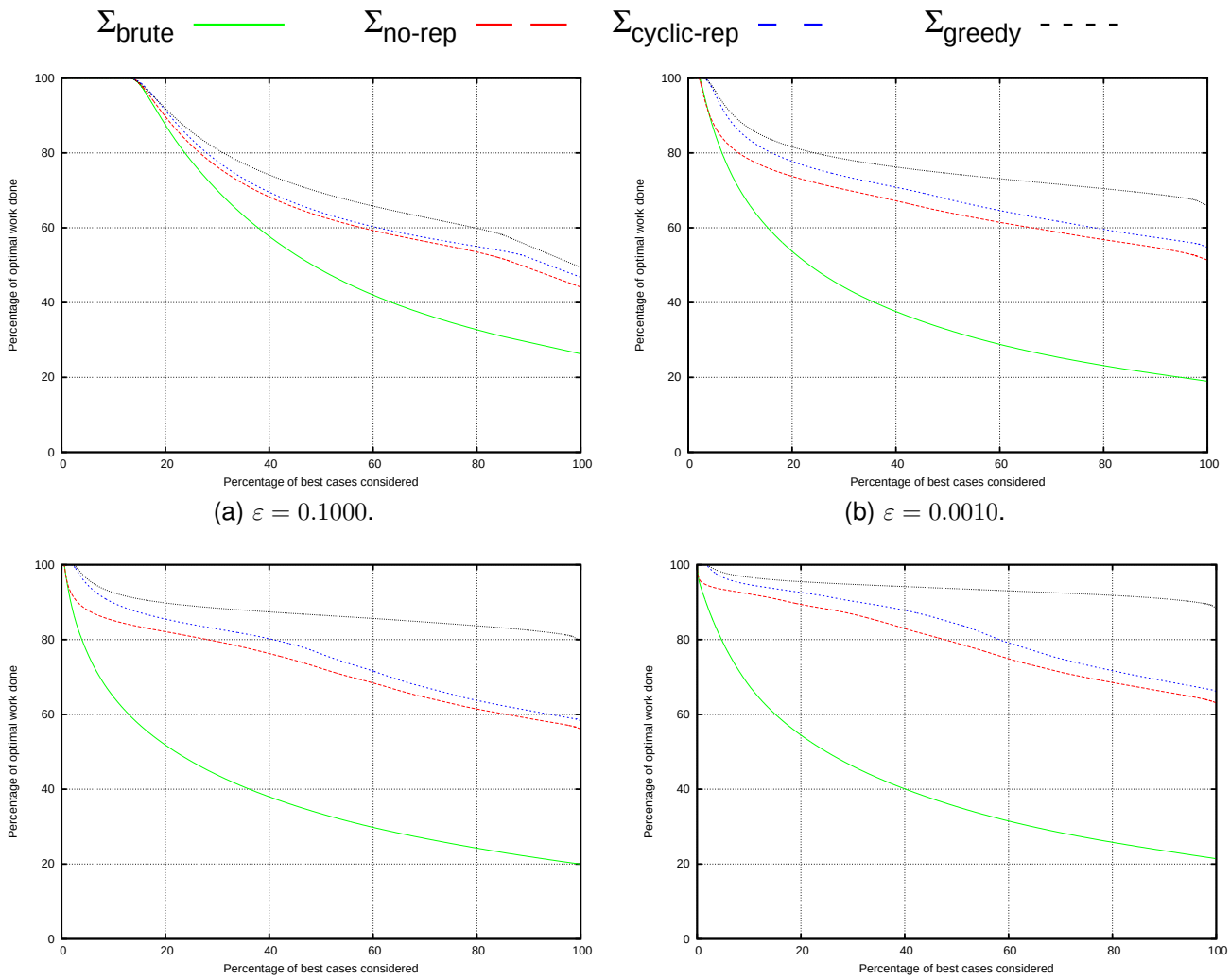
(b) 10 computers.

(c) 25 computers.

(d) 50 computers.

(e) 100 computers.

Fig. 60. Experiments using different numbers of computers.



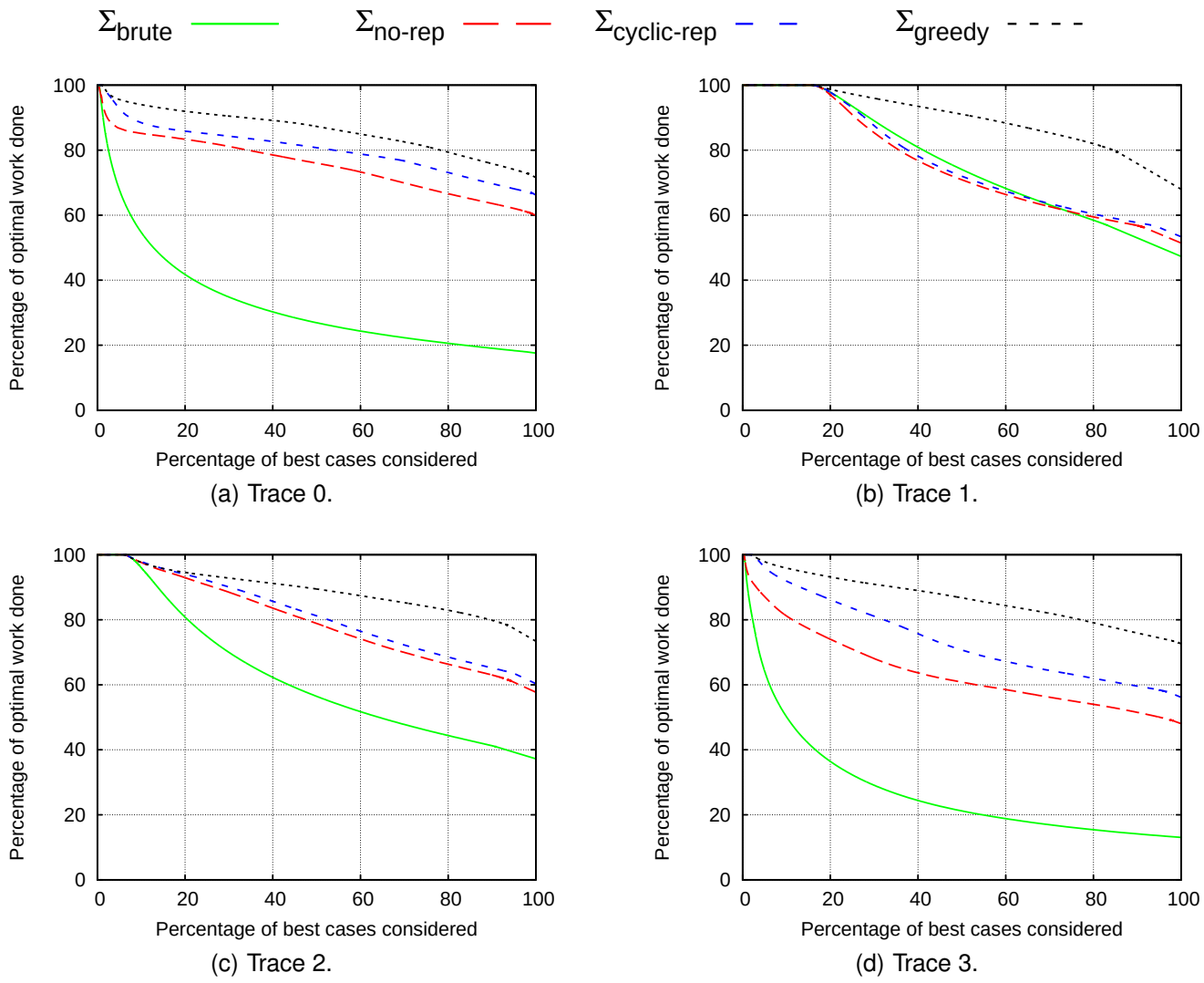
(a) $\varepsilon = 0.1000$.

(b) $\varepsilon = 0.0010$.

(c) $\varepsilon = 0.0010$.

(d) $\varepsilon = 0.0001$.

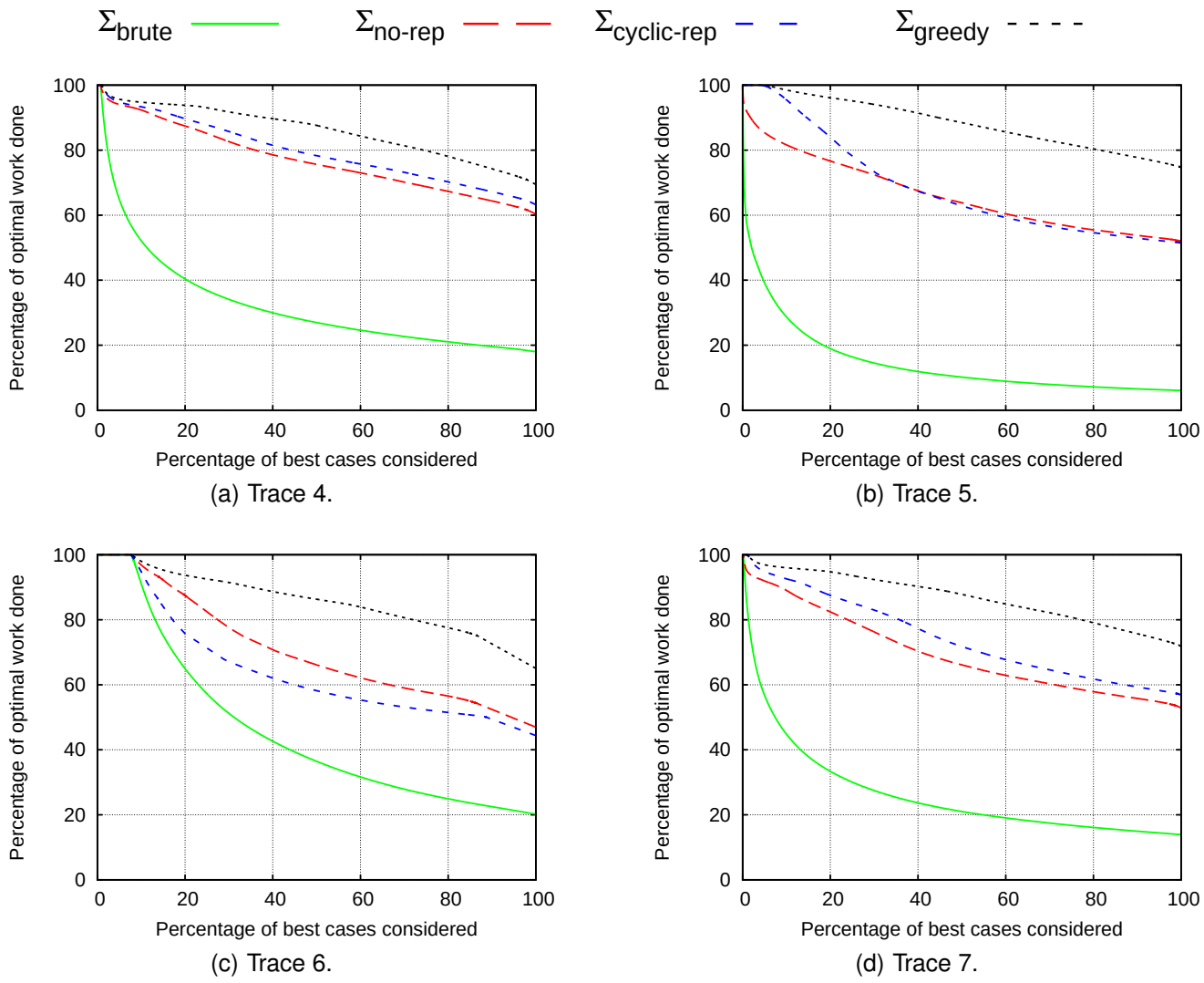
Fig. 61. Experiments using different values of start-up cost.



(a) Trace 0. (b) Trace 1.

(c) Trace 2. (d) Trace 3.

Fig. 62. Experiments using different different traces (a).



(a) Trace 4. (b) Trace 5.

(c) Trace 6. (d) Trace 7.

Fig. 63. Experiments using different different traces (b).