6 Cycle Stealing, Threshold-Based Sharing, and Other 2D-Infinite Chains

In the last section we saw an example of Markov chain that grows unboundedly in more than one dimension, and we introduced a technique, SQA, for solving it, Another place where such 2D-infinite chains come up is in *multi-server systems with dependence*. The most common example in computer science is *cycle stealing*, one of the classical intractable problems in queueing. In this section, we present a new technique, which we call *Dimensionality Reduction for Markov Chains*, and show how this technique can be applied to analyze cycle stealing and many variants thereof.

Cycle stealing problems Consider two queues, each with its own server, as shown in Figure 1(left). Cycle stealing is defined as follows: Name one of the queues the *beneficiary* (B) queue, and the other the *donor* (D) queue. When the donor queue is empty, the donor server can be "donated" to help serve the beneficiary queue. The beneficiary is said to *steal idle cycles* from the donor, hence the name *cycle stealing*. The simplest question one might ask is:

> *How much does a beneficiary's steady-state response time improve when it is allowed to steal cycles?*

The above question is already very difficult to answer, but becomes *more complex* when there are *switching costs* involved. Let T_{switch} be the time required for the donor server to switch to working on the beneficiary jobs, and T_{back} the time for the donor server to switch back to its own jobs. The existence of switching costs implies that a *less aggressive* form of cycle stealing may be preferable, whereby the donor only helps the beneficiary if the donor is *both* idle *and* the beneficiary has at least N_B (threshold) number of beneficiary jobs. Likewise, the donor only returns to working on donor jobs when there are at least N_D (threshold) number of donor jobs. Now we ask:

> *Given switching costs, how should one optimally set thresholds to limit the degree of cycle stealing?*

There are also situations where a *more aggressive* form of cycle stealing is preferable. Consider the situation where beneficiary jobs are shorter than donor jobs. Then it may be globally optimal to allow the donor to help the beneficiary even when the donor server is *not idle*. There are two ways to do this. Under *beneficiary-side control*, the beneficiary sets a (globally-optimal) threshold, N_B, whereby the donor must help the beneficiary whenever there are N_B or more beneficiary jobs, *regardless of whether or not the donor is idle*, see [19, 23]. Alternatively, under *donor-side control*, the donor sets a (globally-optimal) threshold, N_D, where the donor helps the beneficiary whenever there are fewer than N_D donor jobs. Here the question becomes:

> *When more aggressive cycle stealing is globally optimal, should one use beneficiary-side control, or donor-side control? Also, what is the benefit of using multiple thresholds?*

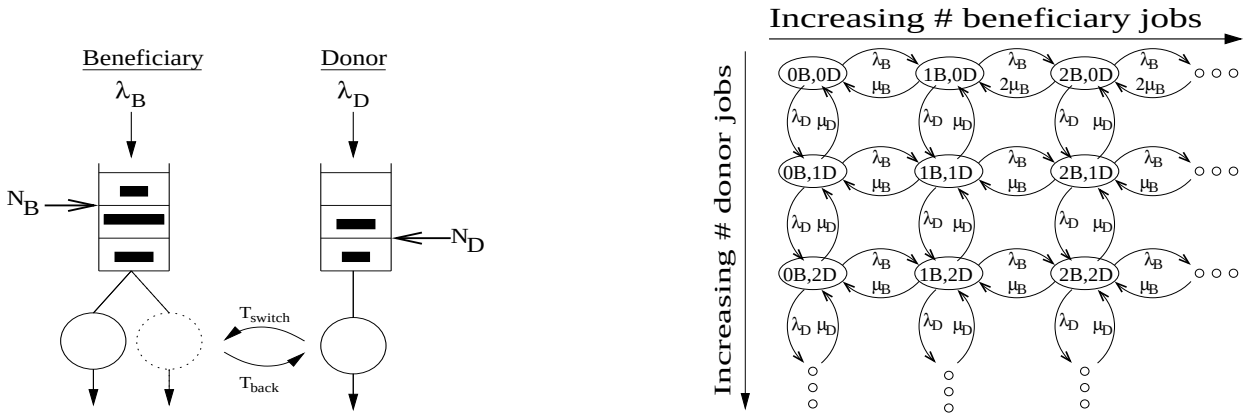

Figure 1: *(Left) 2-server network with cycle stealing. (Right) Associated 2D-infinite continuous-time Markov chain.*

Why cycle stealing is so intractable Even the simplest variant of cycle stealing (without switching costs and without thresholds) is very hard to analyze. The problem is that understanding the beneficiary performance requires analyzing a Markov chain whose state space grows unboundedly (infinitely) in 2 dimensions, see Figure 1, where one dimension tracks the number of beneficiary jobs and the other tracks the number of donor jobs. (In the figure, λ_B and μ_B denote, respectively, the arrival rate, and processing rate, of beneficiary jobs. Likewise for λ_D and μ_D.) While Markov chains which grow infinitely in *one* dimension are usually tractable, 2D-infinite chains are not. Approximations methods tried include: truncating the state space along one of the dimensions [6, 20, 21] which can produce great inaccuracy under high load since portions of the state space are removed; boundary-value methods [4, 11, 3] which are elegant but difficult to evaluate; and heavy-traffic techniques [1, 5] which require assuming high load.

Our approach: Dimensionality Reduction Our approach is quite different from all of the above approaches. The idea is to convert the 2D-infinite chain shown in Figure 1(b) into a 1D-infinite chain, shown in Figure 2, which we can solve. This is achieved without truncation. To do this, we need to introduce a new type of transition into the Markov chain, marked in bold and labeled B_D. Here B_D represents the duration of a *donor busy period*, namely the time from when the donor server has at least one job until it is free. By using busy period transitions, we are capturing everything that the beneficiary needs to know about the donor: whether the donor has 0 jobs (and can help), or whether the donor has at least one job (and can't help), and exactly how long it will be until the donor can again help (the busy period).

The representation in Figure 2(left) is an *exact* representation of the cycle stealing problem, from the perspective of the beneficiary. However it is not yet analyzable because the transition labeled B_D does not have an exponential distribution, as needed for a continuous-time Markov chain. Our approach is to approximate the donor busy period by a phase-type (PH) distribution, consisting of exponential transitions which together allow us to match the moments of the donor busy period. Figure 2(right) shows B_D replaced by a particular PH distribution, which exactly matches the first *three moments* of the donor busy period (we devise optimal moment-matching algorithms in [14, 13, 15]). While our analysis is inherently approximate, in that the busy period distributions are approximated, our approach can be made *as exact as desired* by increasing the number of busy

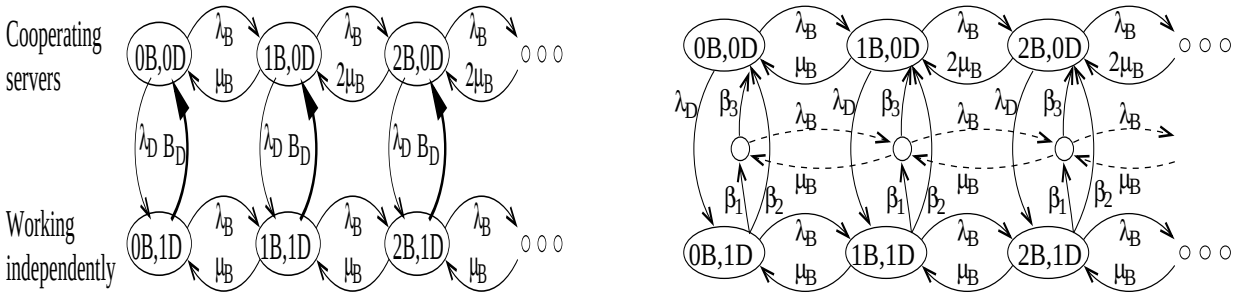

Figure 2: *Illustration of* dimensionality reduction*, reducing the 2D-infinite chain in Figure 1 to a 1D-infinite chain.*

period moments matched. In fact, we have applied our dimensionality reduction technique with matching 3 moments of the busy period to a wide range of open problems, and have achieved accuracy within a couple percent of simulation in every case [10, 9, 22, 2, 17, 18, 7, 16, 8].

Cycle stealing results Our analysis allows us to obtain the first accurate performance numbers (mean and second moment of response time for each job type) for a wide variety of cycle stealing problems. We highlight just a few results below. One interesting question is understanding how the *variability* of the donor workload (specifically, the donor job size distribution) affects the beneficiary under cycle stealing. We find that the beneficiary is only sensitive to the variability in the donor workload when the beneficiary is overloaded or near overload [16]. Other interesting questions have to do with setting beneficiary and donor-side thresholds. We find that when *less-aggressive* cycle stealing is desired, due to *high switching costs*, increasing the donor-side threshold is far more effective at reducing the effects of switching costs than is increasing the beneficiary-side threshold [17]. By contrast, in cases when *more-aggressive* cycle stealing is optimal, using a beneficiary-side threshold to control the donor's help is far more effective with respect to reducing overall mean response time than using a donor-side threshold [18]. However, if we change the metric to *robustness*, rather than response time, then the donor-side threshold is superior [9]. In [18], we propose a solution that achieves *both* excellent mean response time and robustness, by using *two beneficiary-side thresholds*, plus a single donor-side threshold, where the donor-side threshold is used to control which of the two beneficiary-side thresholds gets used. This solution, which we call *Adaptive Dual Threshold*, achieves performance surprisingly close to having an infinite spectrum of thresholds, an idea that has been explored in [12]. Finally, we have also proven cycle stealing to be extremely effective in the context of task assignment for server farms, where short jobs are allowed to steal cycles from long jobs in a non-preemptive fashion [7, 8].

Funding This work has been funded by an NSF Theory grant CCR-0311383 (2003-2006).

References

[1] S. Bell and R. Williams. Dynamic scheduling of a system with two parallel servers in heavy traffic with complete resource pooling: Asymptotic optimality of a continuous review threshold policy. *Annals of Probability*, 11:608–649, 2001.

[2] A. Bhandari, A. Scheller-Wolf, and M. Harchol-Balter. An exact and efficient algorithm for the constrained dynamic operator staffing problem for call centers. *Management Science, to appear*, 2007.

[3] J. Cohen and O. Boxma. *Boundary Value Problems in Queueing System Analysis*. North-Holland Publ. Cy., 1983.

[4] G. Fayolle and R. Iasnogorodski. Two couple processors: the reduction to a Riemann-Hilbert problem. *Zeitschrift fur Wahrscheinlichkeitstheorie und vervandte Gebiete*, 47:325–351, 1979.

[5] R. D. Foley and D. McDonald. Exact asymptotics of a queueing network with a cross-trained server. In *Proceedings of INFORMS Annual Meeting*, pages MD06–2, October 2003.

[6] L. Green. A queueing system with general use and limited use servers. *Operations Research*, 33(1):168–182, 1985.

[7] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. Squillante. Cycle stealing under immediate dispatch task assignment. In *15th ACM Symposium on Parallel Algorithms and Architectures*, pages 274–285, San Diego, CA, June 2003.

[8] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. Squillante. Task assignment with cycle stealing under central queue. In *23rd International Conference on Distributed Computing Systems*, pages 628–637, Providence, RI, May 2003.

[9] M. Harchol-Balter, T. Osogami, and A. Scheller-Wolf. Robustness of threshold policies in a beneficiary-donor model. *Performance Evaluation Review*, 33(2), 2005.

[10] M. Harchol-Balter, T. Osogami, A. Scheller-Wolf, and A. Wierman. Multi-server queueing systems with multiple priority classes. *QUESTA*, 51(3–4):331–360, 2005.

[11] A. Konheim, I. Meilijson, and A. Melkman. Processor-sharing of two parallel lines. *Journal of Applied Probability*, 18:952–956, 1981.

[12] S. Meyn. Sequencing and routing in multiclass queueing networks part i: Feedback regulation. *SIAM Journal on Control Optimization*, 40(3):741–776, 2001.

[13] T. Osogami and M. Harchol-Balter. A closed-form solution for mapping general distributions to minimal PH distributions. In *13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 200–217, 2003.

[14] T. Osogami and M. Harchol-Balter. Necessary and sufficient conditions for representing general distributions by Coxians. In *13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 182–199, 2003.

[15] T. Osogami and M. Harchol-Balter. Closed form solutions for mapping general distributions to quasi-minimal PH distributions. *Performance Evaluation*, 63(6):524–552, 2006. Special Issue for best papers of TOOLS 2003.

[16] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching times and thresholds. In *Proceedings of ACM SIGMETRICS*, pages 184–195, San Diego, CA, June 2003.

[17] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching times and thresholds. *Performance Evaluation*, 61(4):374–369, 2005.

[18] T. Osogami, M. Harchol-Balter, A. Scheller-Wolf, and L. Zhang. Exploring threshold-base policies for load sharing. In *Forty-second Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois, Urbana-Champaign, October 2004.

[19] M. Squillante, C. Xia, D. Yao, and L. Zhang. Threshold-based priority policies for parallel-server systems with affinity scheduling. In *Proceedings of the IEEE American Control Conference*, pages 2992–2999, June 2001.

[20] D. Stanford and W. Grassman. The bilingual server system: A queueing model featuring fully and partially qualified servers. *INFOR*, 31(4):261–277, 1993.

[21] D. Stanford and W. Grassmann. Bilingual server call centers. In D. McDonald and S. Turner, editors, *Analysis of Communication Networks: Call Centers, Traffic and Performance*. American Mathematical Society, 2000.

[22] A. Wierman, T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. How many servers are best in a dual-priority M/PH/k system? *Performance Evaluation*, 2006.

[23] R. Williams. On dynamic scheduling of a parallel server system with complete resource pooling. In D. McDonald and S. Turner, editors, *Analysis of Communication Networks: Call Centers, Traffic and Performance*. American Mathematical Society, 2000.