Variable Annealing Length and Parallelism in Simulated Annealing

Vincent A. Cicirello, Ph.D.
Professor of Computer Science / Behavioral Neuroscience
cicirelv@stockton.edu http://www.cicirello.org/

Introduction

- We introduce a restart schedule of run lengths for an adaptive simulated annealer.
  - Eliminates need to know/predict available time a priori.
- We extend the restart schedule to parallel implementation.
- The variable annealing length restart schedule leads to improved anytime behavior early in the run.
- Discuss experiments with an NP-Hard scheduling problem with sequence-dependent setups.

Background: Simulated Annealing (SA)

- **Annealing:** process of slowly cooling a heated metal.
  - Heating metal allows shaping, while cooling slowly minimizes internal stress / more stable final state.
- **SA:** stochastic search inspired by annealing process.
  - Temperature parameter controls acceptance of neighbors.
    - High temperature early in run = random search
    - Low temperature late in run = stochastic hill climb
    - Cool too quickly = converge too soon to local optima
    - Cool too slowly = excessive run length

Background: SA Annealing Schedules

- Most common annealing schedules:
  - Exponential cooling: $T_{i+1} = \alpha\, T_i$
  - Linear cooling: $T_{i+1} = T_i - \Delta T$
- Boyan's Modified Lam Annealing Schedule (Boyan 1998):
  - Temperature fluctuates up/down to track a theoretical "ideal" neighbor acceptance rate
    - Acceptance rate decreases exponentially from 1.0 (random search) to 0.44 during first 15% of run, and held at 0.44 for next 50% of run.
    - Declines exponentially for last 35% of run (stochastic hill climb).

Related Work: Restarts

- Restarting SA:
  - Many have shown one long run of SA usually outperforms multiple short independent runs.
  - Effective SA restarts likely involves dependent runs.
    - E.g., Sadeh et al (‘97) models expected cost improvement, abandons less promising runs, reanneals other prior runs.
- Restarting other forms of search quite effective:
  - Restarting backtracking CSP search (e.g., Luby schedule)

Related Work: Parallel SA

- Three categories of parallel SA:
  1. Parallel neighbor evaluations
     - E.g., speculative moves (Ludwin & Betz 2011)
  2. Parallel multistart (with dependent runs)
     - E.g., regular intervals sharing best solution among parallel instances (Jha & Menon, 2014)
  3. Optimizing subproblems in parallel
     - E.g., graph partitioning (Rahimian et al 2015)

Approach: Variable Annealing Length

- **Rationale:** long run of SA typically outperforms multiple short runs, but difficult to accurately predict available time.
- **Variable Annealing Length (VAL):**
  - Multistart SA with increasing run lengths.
  - The length of restart r, in number of SA evaluations is:

$$\text{MaxEvals}(r) = 1000 * 2^r$$

  - The multistart SA follows the sequence of run lengths: {1000, 2000, 4000, 8000, … }

Approach: Parallel VAL, version 0

- **Parallel Variable Annealing Length, v0 (P-VAL-0):**
  - Assume N parallel instances of SA: $\{SA_0, SA_1, \ldots, SA_{N-1}\}$
  - The length of restart r of instance SA_i is:

    $$\mathrm{MaxEvals}_i(r) = 1000 * 2^{i+r*N}$$

  - When N=1, P-VAL-0 reduces to VAL.
  - Example, when N=3:
    - Instance 0 follows run lengths: {1000, 8000, 64000, … }
    - Instance 1 follows run lengths: {2000, 16000, 128000, … }
    - Instance 2 follows run lengths: {4000, 32000, 256000, … }

Approach: P-VAL-0's flaw

- P-VAL-0 is flawed:
  - Assuming long run superior to multiple short runs, benefit of parallelization is from completing long runs earlier.
  - As # parallel instances goes to infinity, the longest run completed by P-VAL-0 finishes twice as early as VAL.
    - For N=4 parallel instances, the longest run completed by P-VAL-0 finishes 1.875 times as early as VAL.
    - For N=8 parallel instances, … finishes 1.992 times as early.
    - We hit the limiting behavior with relatively few parallel instances.

Approach: Parallel VAL

- **Parallel Variable Annealing Length (P-VAL):**
  - Assume N parallel instances of SA: $\{SA_0, SA_1, \ldots, SA_{N-1}\}$
  - The length of restart r of instance SA_i is:

$$\text{MaxEvals}_i(r) = 1000 \; * \; 2^{(i \bmod 4) + r * \min(N,4)}$$

  - When N ≤ 4, P-VAL = P-VAL-0; but when N > 4:
    - Instances {0, 4, 8, …} have run lengths: {1000, 16000, 256000, ...}
    - Instances {1, 5, 9, …} have run lengths: {2000, 32000, 512000, ...}
    - Instances {2, 6, 10, …} have run lengths: {4000, 64000, 1024000, ...}
    - Instances {3, 7, 11, …} have run lengths: {8000, 128000, 2048000, ...}

Experiments

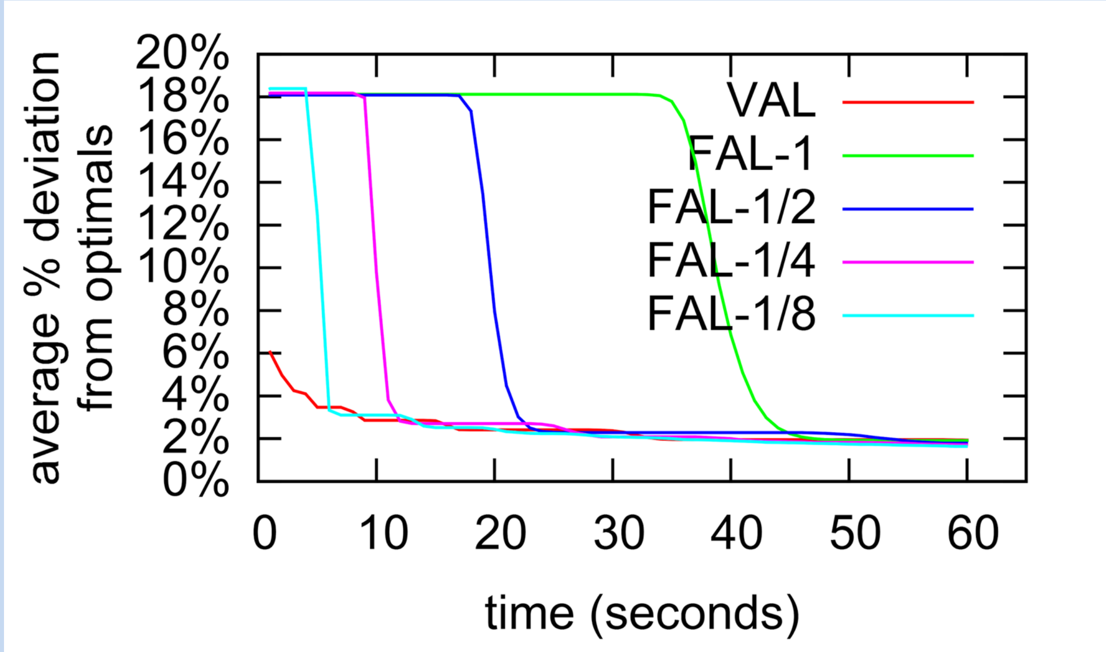
- **Problem:** Scheduling with sequence-dependent setups, minimizing weighted tardiness
  - Used common benchmark set
  - Best exact solver, dynamic programming, > 2 weeks CPU time solving hardest instances. (Tanaka & Araki, 2013)
  - Variety of algorithms applied to problem: neighborhood search (Liao et al, 2012), iterated local search (Xu et al 2014), ACO (Liao & Juan 2007), among others

Experiments

- **Platform:** Ubuntu 14.04 server, 2 Xeon L5520 quad-core (2.27GHz), 32GB; Java 8, Java HotSpot 64-bit Server VM
- Sequential (N=1) and parallel (N=4, N=8) experiments.
- For each algorithm, 10 runs on each of 120 instances, logging best solution at 1 second intervals over 60 s.
- Compare VAL, P-VAL-0, and P-VAL to the following:
  - Fixed annealing length (FAL-x) of x of total run, with restarts
    - E.g., FAL-1=one long run, FAL-½=run length half total time
  - Parallel fixed annealing length (P-FAL-x)

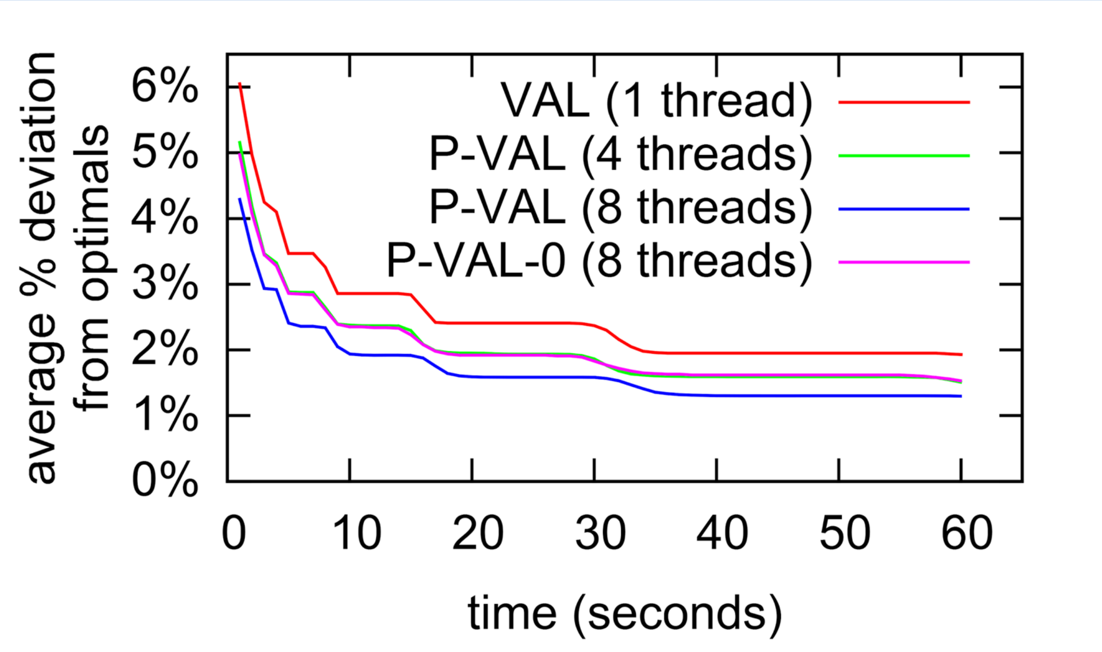
Sequential Results

- VAL dominates fixed annealing length early.
- No significant difference, at end of run (last 12s), between VAL and FAL-1 ($p>0.35$)
- Not visually evident: Single long fixed length does outperform restarts of short fixed length at end.



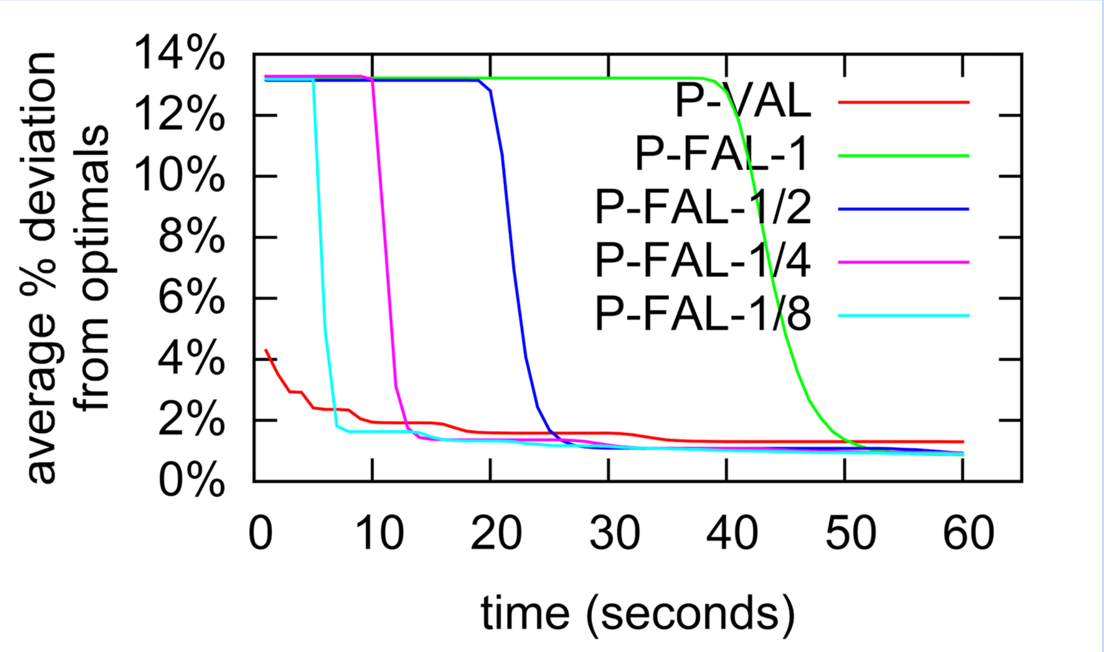
P-VAL vs P-VAL-0

- P-VAL-0 with N>4 parallel runs doesn’t improve performance
  - e.g., P-VAL-0 with N=8 no better than N=4 (green/pink)
- P-VAL continues to see performance gains for N>4 parallel runs.



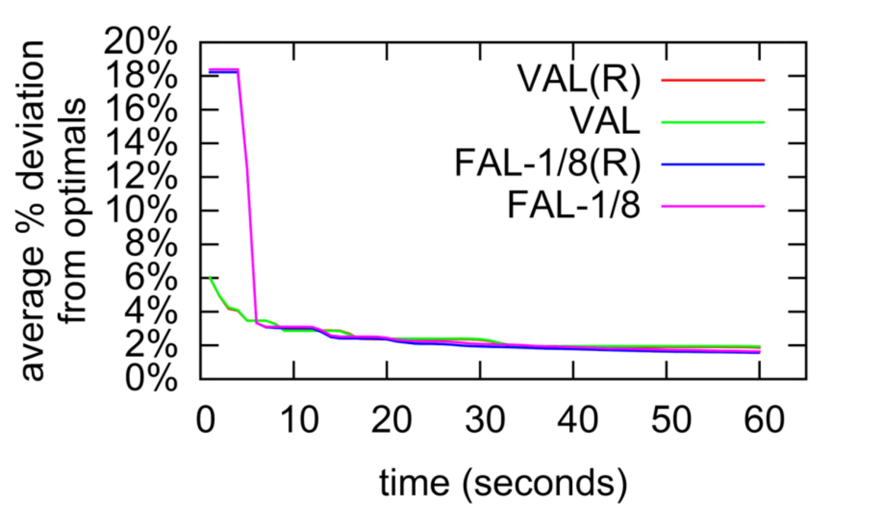
Parallel Results

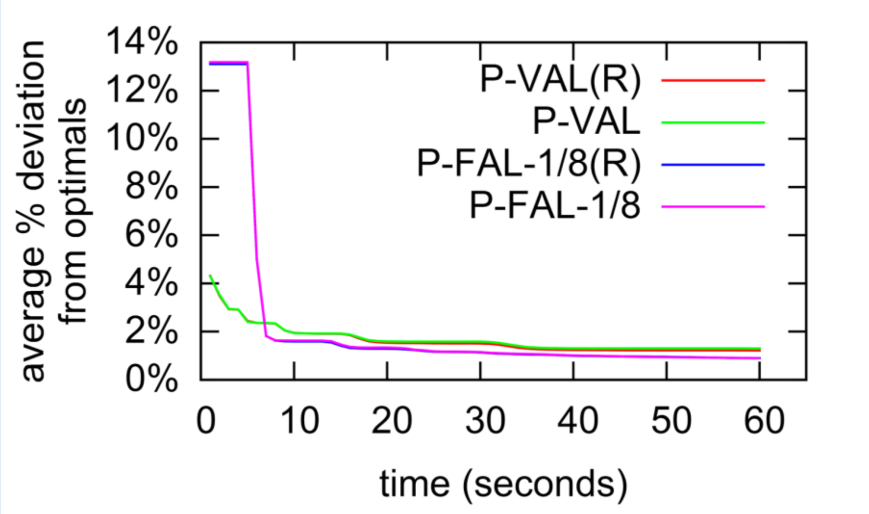
- Results with N=8
- Unlike sequential case, P-VAL does not approx. performance of fixed length restarts at run end.
- P-VAL dominates approx. 80% of P-FAL's first run
  - For 48s vs P-FAL-1
  - For 24s vs P-FAL-½
  - For 12s vs P-FAL-¼
  - For 6s vs P-FAL-⅛



Reannealing vs Independent Runs

- All results so far are for independent runs.
- Also considered restarts that reanneal the current best of run solution
  - Including across parallel runs.
- Sequential results:
  - No significant difference for VAL with reannealing vs independent runs.
  - Same is true for FAL-⅛ with reannealing vs independent runs.
  - Same is true in parallel





Conclusions

- Proposed a multistart SA, with variable annealing length
  - Eliminates need to know/predict available time for run
    - Issue not limited to Modified Lam (e.g., common exponential schedule becomes stochastic hill climb too soon if α too low).
  - Short early runs quickly find “good” solution, and increasing run length approximates final performance of long SA runs.
- Proposed parallel implementation
- Long fixed length runs better at end of run, but variable length restarts exhibits stronger anytime performance.

Questions