Integer Boltzmann Machines for Future Generation Digital Annealing Units

Mohammad Bagherbeik[1], Kouichi Kanda[2], Hirotaka Tamura[3], Ali Sheikholeslami[1]

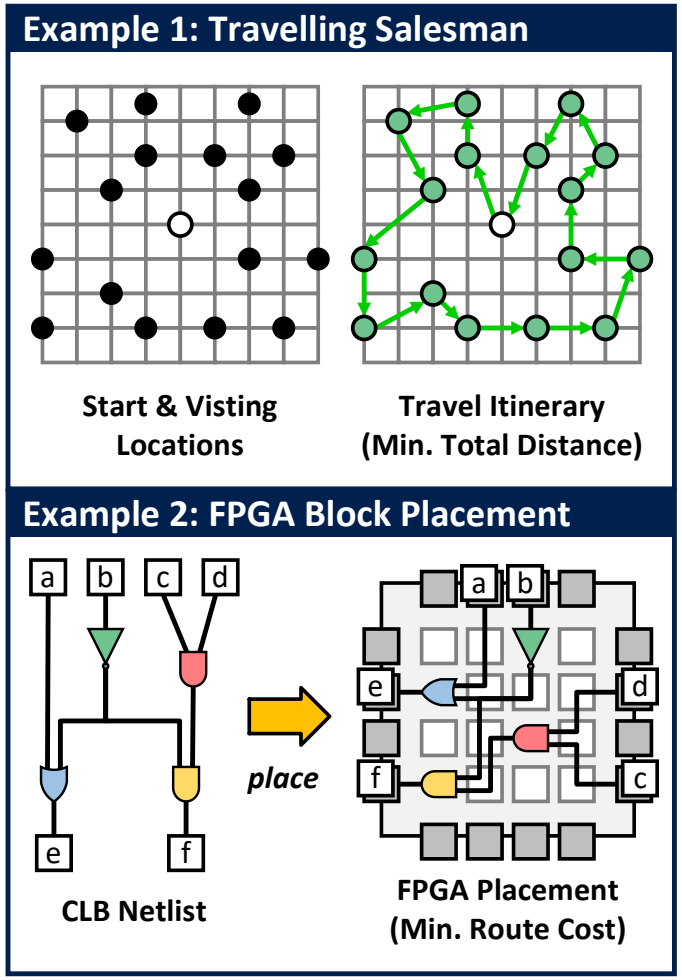
[1]Dept. of Electrical & Computer Engineering, University of Toronto, Canada
[2]Fujitsu Limited, Kawasaki, Japan
[3]DXR Laboratories, Kawasaki, Japan

Adiabatic Quantum Computing Conference 2021

Outline

- **Motivation and Background**
  - Quadratic Assignment Problems
  - Local Search Algorithms
- Integer Boltzmann Machine
- Parallel Tempering Solver Design
  - Algorithm and SIMD
  - Load Balancing
- Benchmark Results
- Conclusion

Integer Assignment Problems

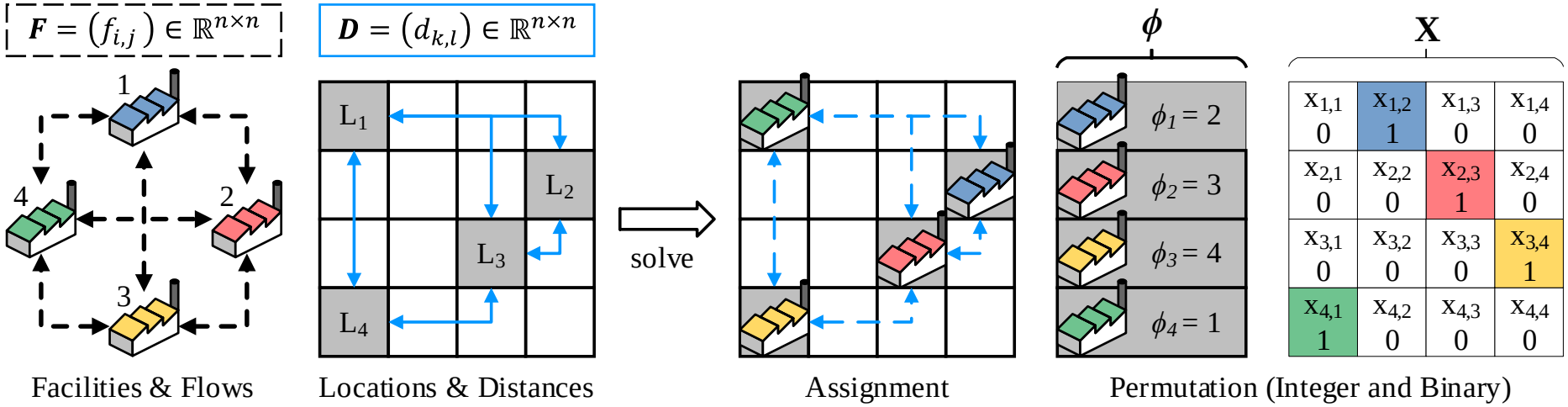


- Class of NP-hard combinatorial optimization problems with integer variables
  - Quadratic Assignment Problem (QAP) [1]
- Simple formulation but difficult to solve even with modern CPU and GPU hardware

Goal

- Create an integer Boltzmann Machine variant to accelerate search for QAP
- Presentation covers formulation for symmetric, bias-less QAPs but concepts are directly transferrable to asymmetric + biased problems

QAP Formulation



- Given n facilities and n locations
- Find a permutation $\boldsymbol{\phi} = (\phi_i) \in \boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ is set of all possible permutations
  - $\boldsymbol{\phi}$ assigns each facility i, to a unique location ϕ_i such that the total cost of the assignment (1) is minimized:

$$C(\boldsymbol{\phi}) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j} d_{\phi_i,\phi_j} \quad (1)$$

$$x_{i,j} = \begin{cases} 1, & if\ \phi_i = 1 \\ 0, & otherwise \end{cases}, \mathbf{X} = (\mathrm{x_{i,j}}) \in \{0,1\}^{n \times n} \quad (2)$$

Local Search

1. Propose new states as perturbations of the current state
   - Exchange locations of 2 facilities

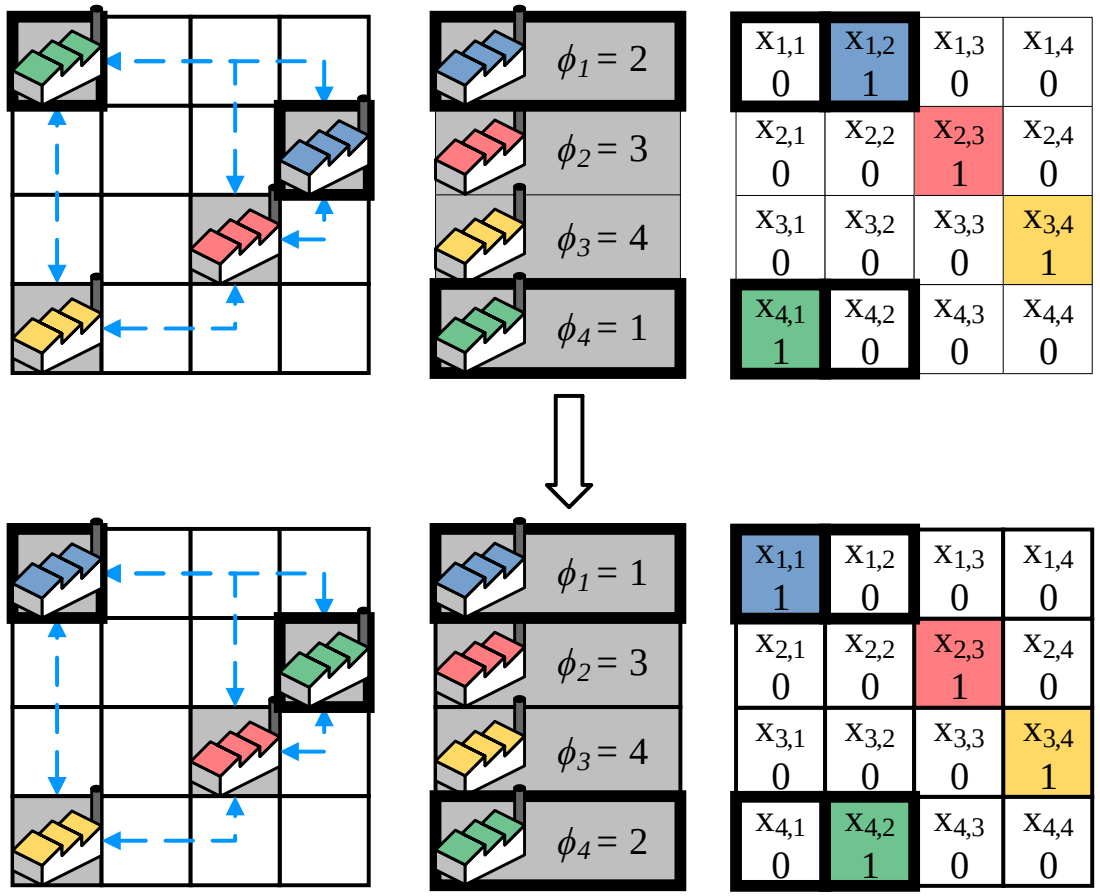
2. Calculate difference in cost function (ΔC) between previous and proposed state

3. Decide whether to accept proposed state based on some criterion
   - Stochastic Approach (e.g. Simulated Annealing):

     Use a varying temperature variable (T) that is used to inject randomness into acceptance of moves via Metropolis Hastings acceptance probability generated using (3)
     - Proposal Acceptance Rate (PAR)
     - $T \to \infty, PAR \to 100\%$, all proposals are accepted regardless of ΔC value
     - $T \to 0, PAR \to 0\%$, search turns greedy

$$P_{acc} = \min\{1, \exp\left(-\frac{\Delta C}{T}\right)\} \qquad (3)$$

Local Search: QAP Exchange



- Exchange locations of two facilities a, b
- Change in cost calculated via (4)
- Implemented in software via simple loop(s) in $\mathcal{O}(n)$ time

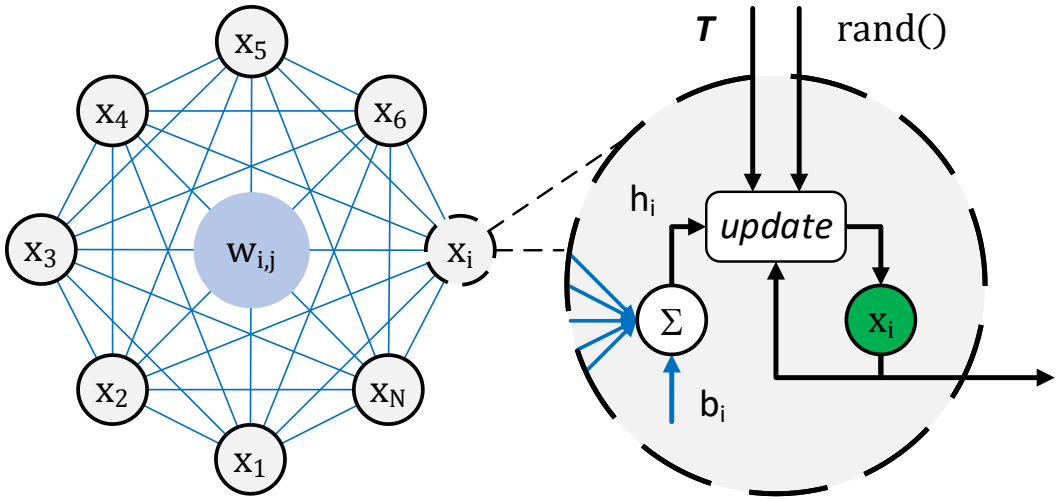
$$\Delta C_{ex} = \sum_{i=1, i \neq a,b}^{n} 2(f_{b,i} - f_{a,i})(d_{\phi_a,\phi_i} - d_{\phi_b,\phi_i}) \quad (4)$$

Outline

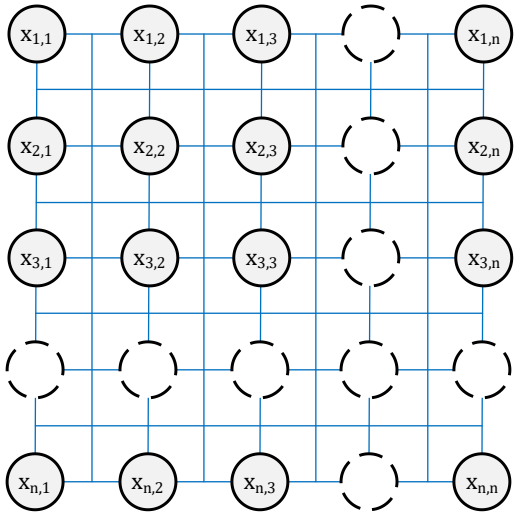
- Motivation and Background
  - Quadratic Assignment Problems
  - Local Search Algorithms
- **Integer Boltzmann Machine**
- Parallel Tempering Solver Design
  - Algorithm and SIMD
  - Load Balancing
- Benchmark Results
- Conclusion

Boltzmann Machines

1D – N neurons

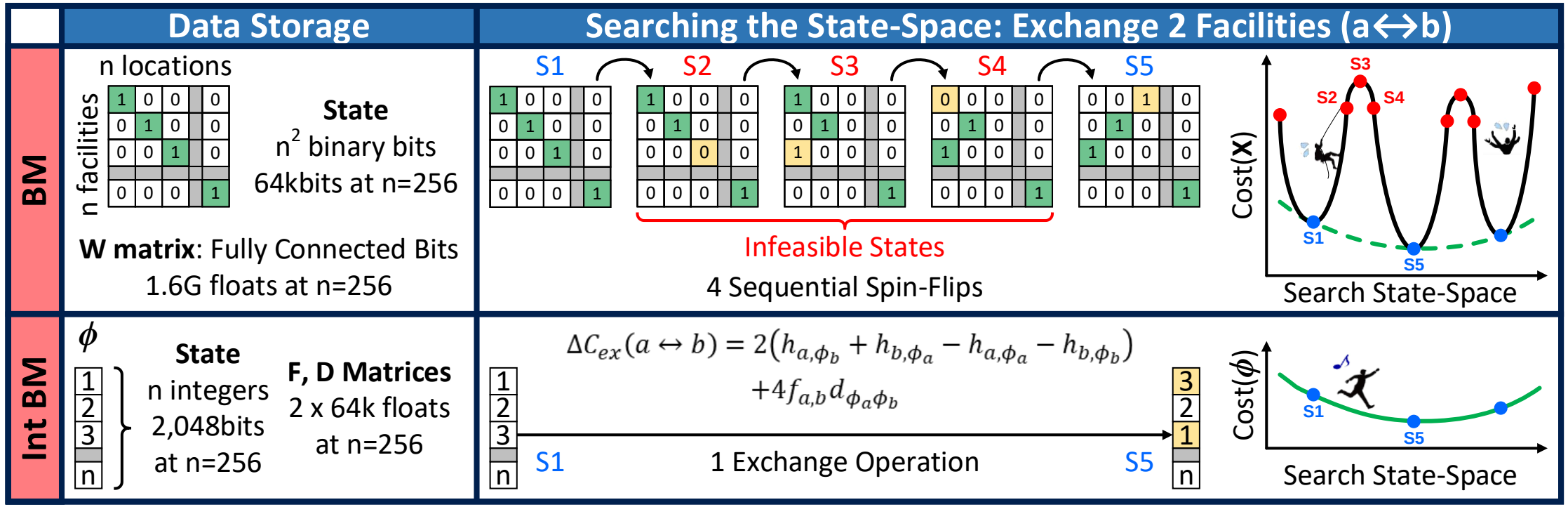


2D – $n \times n$ neurons



State Cost/ Energy	$C(\mathbf{X}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}x_i x_j - \sum_{i=1}^{N} b_i x_i$	(5)	$C(\mathbf{X}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} w_{i,j;k,l}x_{i,j}x_{k,l} - \sum_{i=1}^{n}\sum_{j=1}^{n} b_{i,j}x_{i,j}$	(7)
Neuron Local Fields	$h_i(\mathbf{X}) = \sum_{j=1}^{N} w_{i,j}x_i + b_i$	(6)	$h_{i,j}(\mathbf{X}) = \sum_{k=1}^{n}\sum_{l=1}^{n} w_{i,j;k,l}x_{k,l} + b_{i,j}$	(8)

BM vs Integer BM for QAP [2,3]

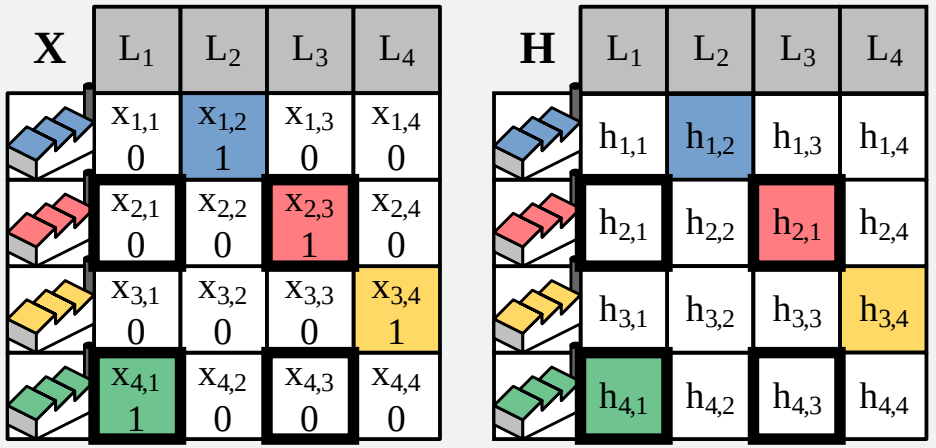


- 2D BM requires penalty terms to enforce one-hot constraints on rows/columns of **X**
- Using an Integer BM with intrinsic exchange moves (4-bit-flip equivalent) simplifies problem formulation and speeds up search, significantly reduces memory reqs.

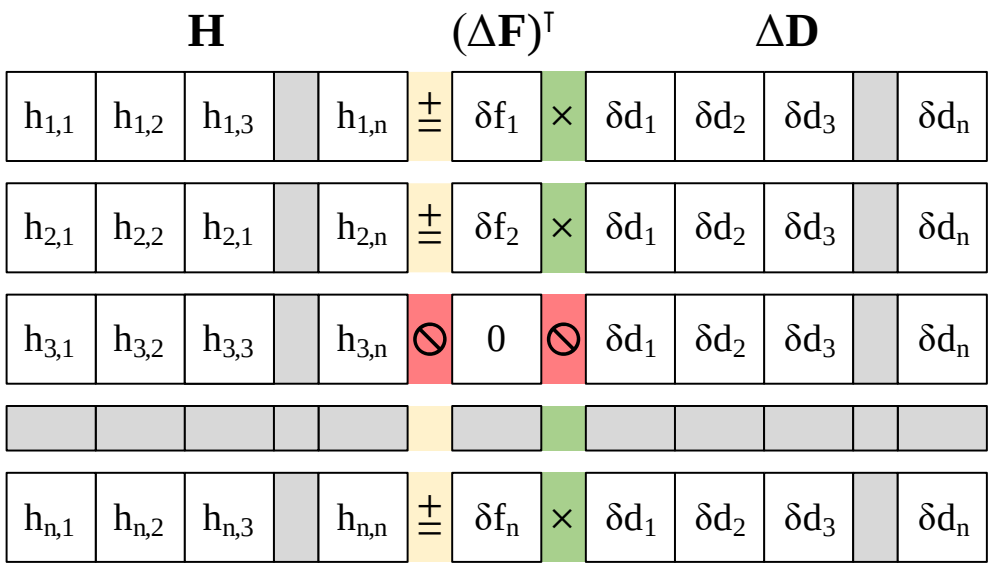
Int BM QAP Formulation

Notes	Formulation	#
• Remove conditions on the summation loop iterator	$\Delta C_{ex} = \sum_{i=1, i\neq a,b}^{n} 2(f_{b,i} - f_{a,i})(d_{\phi_a,\phi_i} - d_{\phi_b,\phi_i})$	(4)
• Introduces extra terms within the sum • Compensate via additional terms	$\Delta C_{ex} = \sum_{i=1}^{n} 2(f_{b,i} - f_{a,i})(d_{\phi_a,\phi_i} - d_{\phi_b,\phi_i}) + 4f_{a,b}d_{\psi_a\psi_b}$	(9)
• Create vector of flow row differences	$\Delta_a^b \mathbf{F} = \mathbf{F}_{b,*} - \mathbf{F}_{a,*}$	(10)
• Create vector of permutation ordered distance row differences	$\Delta_{\phi_b}^{\phi_a} \mathbf{D}^{\mathbf{X}} = \left(\mathbf{D}_{\phi_a,*} - \mathbf{D}_{\phi_b,*}\right)\mathbf{X}^{\intercal}$	(11)
• Reformulate using a dot product	$\Delta C_{ex} = 2\Delta_a^b \mathbf{F} \cdot \Delta_{\phi_b}^{\phi_a} \mathbf{D}^{\mathbf{X}} + 4f_{a,b}d_{\phi_a\phi_b}$	(12)

Integer BM: Local Fields

Notes	Formulation	#
• Generate cache values at start of search in $\mathcal{O}(n^3)$ time	$\mathbf{H} = \mathbf{F}(\mathbf{D}^{\mathsf{T}}\mathbf{X}) = (h_{r,c}) \in \mathbb{R}^{n \times n}$	(13)
• Elements in $\mathbf{H}$ correspond to bits in $\mathbf{X}$ • Element $h_{r,c}$ is cost of assigning a facility r to location c		
• Reformulate dot product as sum of 4 $\mathbf{H}$ elements taking $\mathcal{O}(1)$ time	$\Delta_a^b \mathbf{F} \cdot \Delta_{\phi_b}^{\phi_a} \mathbf{D}^{\mathbf{X}} = h_{a,\phi_b} + h_{b,\phi_a} - h_{a,\phi_a} - h_{b,\phi_b}$	(14)
$\Delta C_{ex} = 2\Delta_a^b \mathbf{F} \cdot \Delta_{\phi_b}^{\phi_a} \mathbf{D}^{\mathbf{X}} + 4 f_{a,b} d_{\phi_a \phi_b}$ →	$\Delta C_{ex} = 2(h_{a,\phi_b} + h_{b,\phi_a} - h_{a,\phi_a} - h_{b,\phi_b}) + 4 f_{a,b} d_{\phi_a \phi_b}$	(15)

Int BM: Local Field Updates



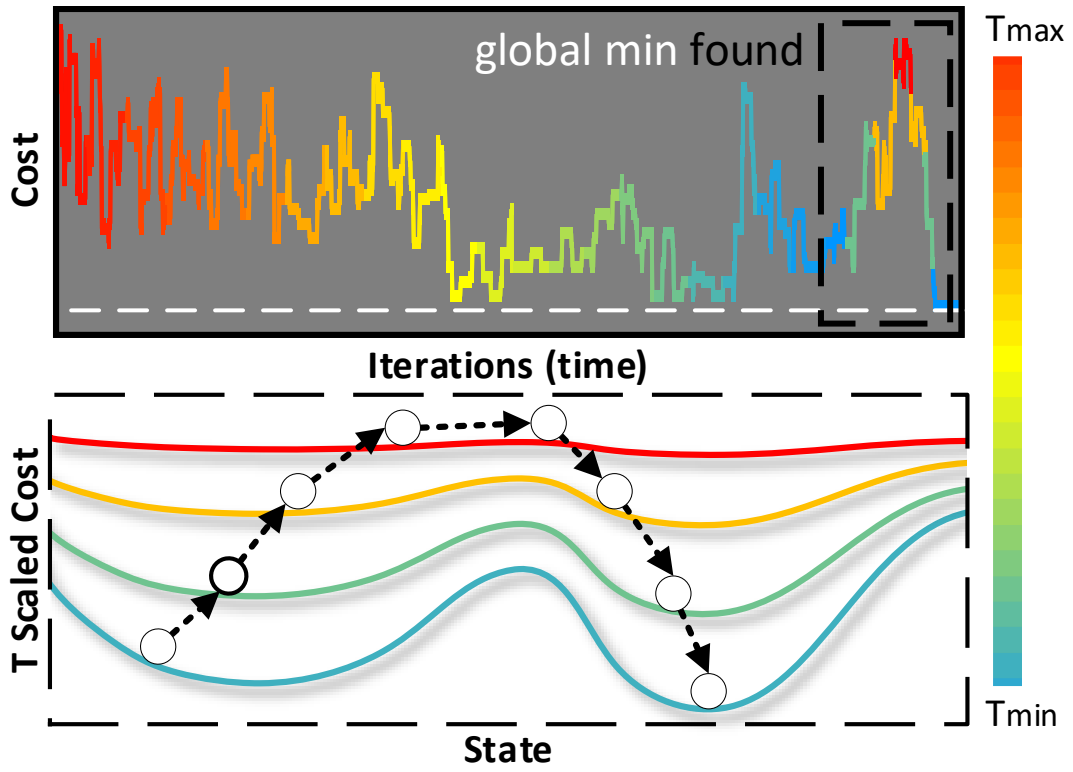
$$\mathbf{H} \mathrel{+}= \left(\Delta_a^b \mathbf{F}\right)^{\intercal} \otimes \Delta_{\phi_b}^{\phi_a} \mathbf{D} \qquad (16)$$

- Update local fields 1 row at a time using floating point fused-multiply-adds
  - $\mathcal{O}(n^2)$
- Can skip rows corresponding to 0 elements in $\Delta\mathbf{F}$
- Significant speed-ups if $\mathbf{F}$ is sparse or has a particular pattern that leads to sparse $\Delta\mathbf{F}$ vectors

Outline

- Motivation and Background
  - Quadratic Assignment Problems
  - Local Search Algorithms
- Integer Boltzmann Machine
- **Parallel Tempering Solver Design**
  - Algorithm and SIMD
  - Load Balancing
- Benchmark Results
- Conclusion

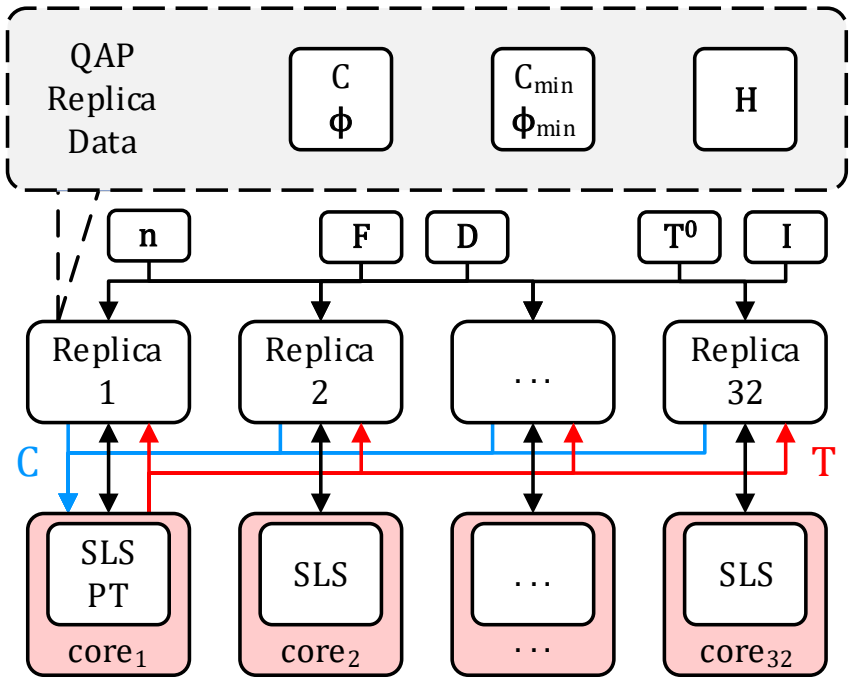
Parallel Tempering



- Uses M unique search-states/replicas with unique temperatures, $T = (T_k) \in R^M$
- Replicas perform Local Search at their assigned temperature
  - Can swap temperatures with other replicas with probability calculated via (17)
  - Allows for an escape mechanism from local minima

$$SAP = min\left\{1, \exp\left(\left(\frac{1}{T_k} - \frac{1}{T_{k+1}}\right)(C_k - C_{k+1})\right)\right\} \quad (17)$$

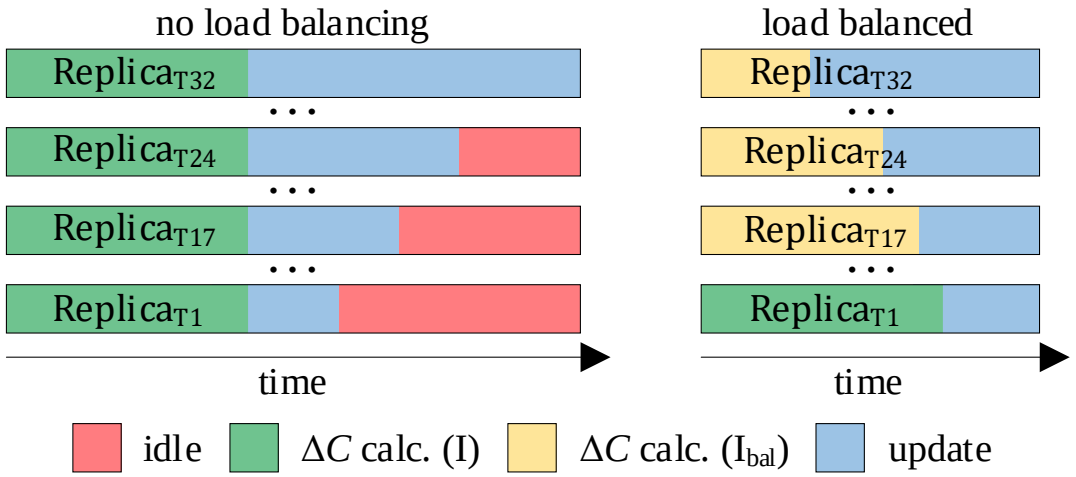
Parallel Tempering Solver System



32 Replica PT Engine

- Experimental setup:
  - 64-Core AMD 3990X CPU with AVX2 support
  - Coded in C++ and compiled with GCC-9
- Float32 used to store $\mathbf{F}, \mathbf{D}, \mathbf{H}$
  - Allows for Fused-Multiply-Add instructions
- Each engine uses 32 replicas in combination with a PT controller
  - Each replica is run on a dedicated CPU core
  - 2 engines are run in parallel to utilize all 64 cores
  - Runs until a target cost is found or a 5-minute time-out limit is reached

Load Balancing Threads



- Varying proposal acceptance rates of replicas at different temperatures causes thread imbalance when using Parallel Tempering
- Track average time-per-iteration at each temperature and scale iterations at each temperature to try to equalize thread run-times

Outline

- Motivation and Background
  - Quadratic Assignment Problems
  - Local Search Algorithms
- Integer Boltzmann Machine
- Parallel Tempering Solver Design
  - Algorithm and SIMD
  - Load Balancing
- **Benchmark Results**
- Conclusion

QAP Benchmark Results

TABLE I: QAP TtS Across Solvers (values in seconds)

Instance	ParEOTS	Int BM + PT	Speed-Up
dre42	0.7	0.10 ± 0.01	7.00×
dre56	5.6	0.99 ± 0.07	5.66×
dre72	26.0	5.23 ± 0.31	4.97×
Inst50	17.0	3.58 ± 0.27	4.75×
Inst60	67.0	2.00 ± 0.16	33.5×
Inst70	127.0	6.24 ± 0.98	20.35×
Inst80	116.0	19.36 ± 1.75	5.99×
sko81	22.4	0.13 ± 0.01	172.31×
sko90	92.0	0.26 ± 0.01	353.85×
sko100a	69.0	0.27 ± 0.01	255.56×
sko100b	45.0	0.19 ± 0.01	236.84×
sko100c	56.0	0.29 ± 0.02	193.1×
sko100d	37.0	0.38 ± 0.02	97.37×
sko100e	47.0	0.21 ± 0.01	223.81×
sko100f	57.0	0.24 ± 0.01	237.5×
tai60b	4.6	0.14 ± 0.01	32.86×
tai80b	53.0	0.43 ± 0.01	123.26×
tai100b	71.0	0.36 ± 0.02	197.22×
wil100	97.0	0.46 ± 0.02	210.87×
Geometric Mean Speedup			**57.09×**

- Avg. Time-to-Solution (TtS) across 100 independent, randomly seeded, runs with 99% confidence interval for 19 difficult QAP instances
  - Instances from [4,5,6]
- ParEOTS [7]: Parallel Extremal Optimization + Tabu Search
  - Best performing published QAP solver
  - 128 cores on CPU cluster 8 x 16–core AMD 6376 CPU
  - TtS reported as average across 10 runs with 5-minute time-out
- 57x faster than ParEOTS on average
  - Max speed-up of 353x

Conclusion

- Extended BMs to support for integer variables with one-hot constraints
  - Remove need for penalty tuning on user side
- Added support for weight-generation using submatrices for problem topologies such as QAP
  - Larger problem support due to lower memory requirements
- Integer BM based CPU solution displays performance that is orders of magnitude faster than competing solvers across difficult QAP instances

Future Work

- Accelerate search relative to high-performance multi-core CPU system through dedicated hardware

Acknowledgements

The authors would like to thank Fujitsu Ltd. and Fujitsu Consulting (Canada) Inc. for providing financial support and technical expertise on this research and to CMC Microsystems for access to CAD tools used in this research

Thank You!

Looking forward to your questions at the live session

References

- [1] Lawler, E.L.: The quadratic assignment problem. Manage. Sci. 9(4), 586–599 (1963).
- [2] M. Bagherbeik, P. Ashtari, S. F. Mousavi, K. Kanda, H. Tamura, and A. Sheikholeslami, "A permutational boltzmann machine with parallel tempering for solving combinatorial optimization problems," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2020, pp. 317–331
- [3] M. Bagherbeik and A. Sheikholeslami, "Caching and vectorization schemes to accelerate local search algorithms for assignment problems," in *IEEE Congress on Evolutionary Computation (to appear)*, 2021.
- [4] Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIP-a quadratic assignment problem library. J. Global Optim. 10(4), 391–403 (1997).
- [5] Palubeckis, G.: An algorithm for construction of test cases for the quadratic assignment problem. Informatica Lith. Acad. Sci. 11, 281–296 (2000)
- [6] Drezner, Z., Hahn, P.M., Taillard, E.D.: Recent advances for the quadratic assign- ´ ment problem with special emphasis on instances that are difficult for metaheuristic methods. Ann. Oper. Res. 139(1), 65–94 (2005).
- [7] Munera, Danny, Daniel Diaz, and Salvador Abreu. "Hybridization as cooperative parallelism for the quadratic assignment problem." *International Workshop on Hybrid Metaheuristics*. Springer, Cham, 2016.