Challenges in solving Large Scale Combinatorial Optimization Problems

HEMERA – Challenge COPs

Leaders: B. Derbel, Nouredine Melab

Participants: Trong-Tuan Vu (Hemera), Asim Ali (Hemera), Mathieu Djamaï (U. Lille 1), Mohand Mezmaz (U. Mons)

DOLPHIN – INRIA Lille Nord Europe

HEMERA – Challenge COPs

Combinatorial Optimization

- Applications in: Logistics, Energy, Clouds, Telecommunication, etc.
- NP-Hard problems
- Resolution methods are computing intensive

Distributed Computing

- Aggregated computing resources
- New architectures and facilities, e.g. Clouds
- Impressive computing power (in theory)

HEMERA – Challenge COPs

Combinatorial Optimization

- Applications in: Logistics, Energy, Clouds, Telecommunication, etc.
- NP-Hard problems
- Resolution methods are computing intensive

Distributed Computing

- Aggregated computing resources
- New architectures and facilities, e.g. Clouds
- Impressive computing power (in theory)

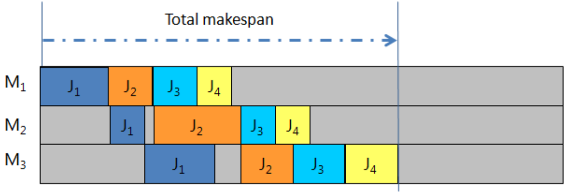
Challenge

- Solve large scale Combinatorial Optimization Problems (COPs) using huge amount of computational resources.

Outline

1. Parallel Branch-and-Bound (B&B) for Permutational FSP
   - **Dynamic Load Balancing**
2. Discussion and other related results
   - Large scale P2P distributed computing in COPs
   - Heterogenous computing in COPs
3. Conclusion

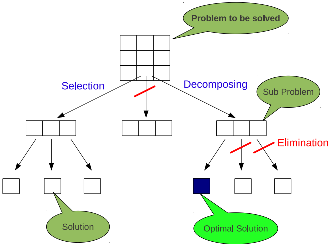
Parallel B&B for Permutational FSP



Branch and Bound Tree Search

B&B

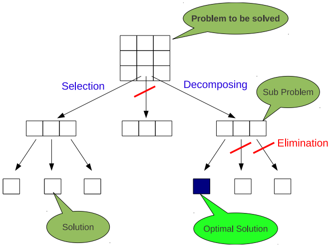
- Decomposing: split a problem into sub-problems
- Bounding: compute lower bound
- Elimination: eliminate bad branches
- Selection: chose next node to explore



Branch and Bound Tree Search

B&B

- Decomposing: split a problem into sub-problems
- Bounding: compute lower bound
- Elimination: eliminate bad branches
- Selection: chose next node to explore



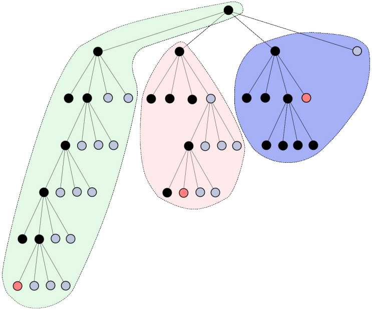
Parallel B&B

- Process B&B subtrees distributively in parallel
- Communicate the best found solution

Parallel B&B

Parallel Modeling

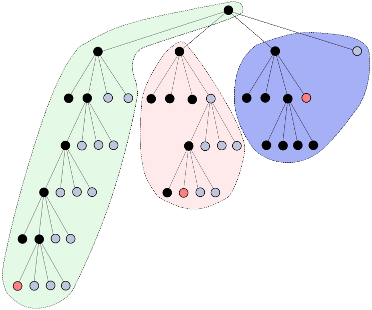
- Perform a parallel tree traversal
  - The tree is generated at runtime
  - Unpredictable and unbalanced shape
- Unbalanced Tree Search (UTS) [Dinan et al., 2008]



Parallel B&B

Parallel Modeling

- Perform a parallel tree traversal
  - The tree is generated at runtime
  - Unpredictable and unbalanced shape
- Unbalanced Tree Search (UTS) [Dinan et al., 2008]



Main Question

- How to distribute workload over processing units, dynamically, at runtime?

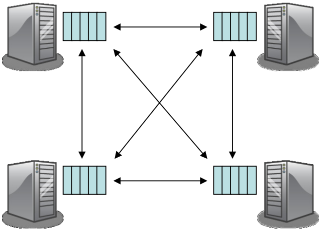
Litterature overview

Combinatorial Optimization (B&B) Community

- Master-Worker [IPDPS'07]
- Hiearchical Master-Worker [FGCS'12, IEEE TC'13]
- B&B specific coding

HPC Community

- Random Work Stealing (application independent)
- Theory:
  - Expected time: $\frac{W}{p} + O(D)$ [Blumofe et al., ACM'99]
- Practice / Applications:
  - Steal-half gives good performance [Dinan et al., SC'09]:
  - Work stealing for multicore systems [Euro-Par'11, PPoPP'13]



Litterature overview

HPC Community

- Random Work Stealing (application independent)
- Theory:
  - Expected time: $\frac{W}{p} + O(D)$ [Blumofe et al., ACM'99]
- Practice / Applications:
  - Steal-half gives good performance [Dinan et al., SC'09]:
  - Work stealing for multicore systems [Euro-Par'11, PPoPP'13]

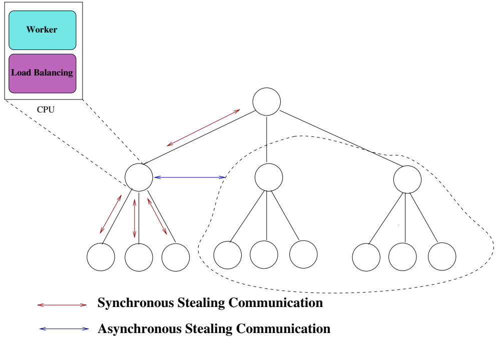
The main issues

1. Where to search work?
   - Location of work is not known
2. What work sharing strategy (steal granularity)?
   - Number of stealing operations

Our Contribution

Overlay-based Load Balancing

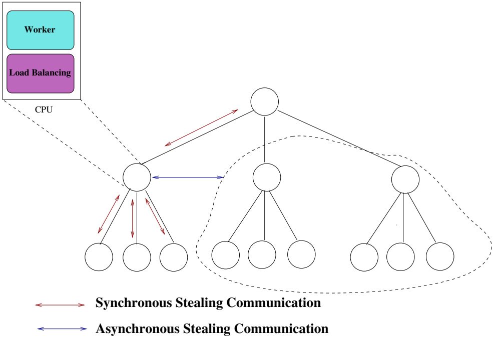
1. Overlay structure and Peer Cooperation
   - Thieves cluster together along a **tree overlay**



Our Contribution

Overlay-based Load Balancing

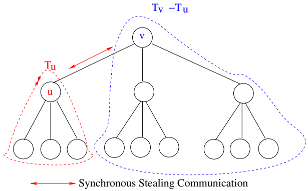
1. Overlay structure and Peer Cooperation
   - Thieves cluster together along a **tree overlay**
2. Adaptive work sharing
   - **Stealing granularity** adapts to peers computing power



Overlay-based granularity

Between close neighbors (synchronous)

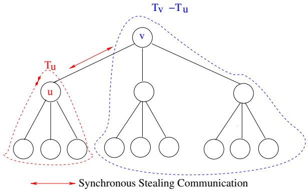
- Children u steals from Parent v: T_u / T_v
- Parent v steals from Children u: $(T_v - T_u) / T_v$



Overlay-based granularity

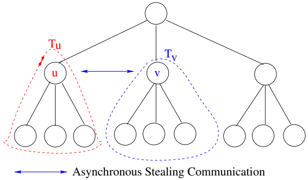
Between close neighbors (synchronous)

- Children u steals from Parent v: T_u/T_v
- Parent v steals from Children u: $(T_v - T_u)/T_v$



Between remote neighbors (asynchronous)

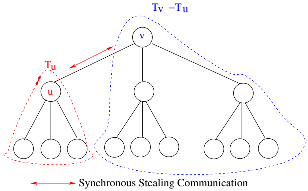
- Peer u steals from peer v: $T_u/(T_u + T_v)$



Overlay-based granularity

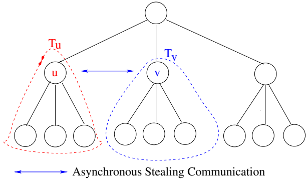
Between close neighbors (synchronous)

- Children u steals from Parent v: T_u/T_v
- Parent v steals from Children u: $(T_v - T_u)/T_v$



Between remote neighbors (asynchronous)

- Peer u steals from peer v: $T_u/(T_u + T_v)$



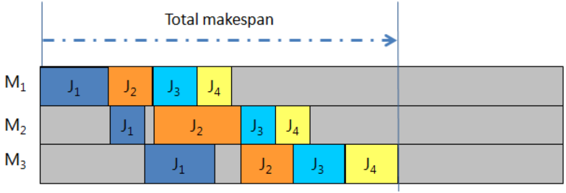
Other technical issues

- Communication of best solution (B&B specific)
- Termination detection

Experimental Validation

Application settings

- Parallel B&B: Taillard' Flowshop Instances (Ta20*20)
  - Permutational FSP: 20 jobs on 20 machines
  - Sequential execution: some hours to some days



Experimental Validation

Application settings

- Parallel B&B: Taillard' Flowshop Instances (Ta20*20)
  - Permutational FSP: 20 jobs on 20 machines
  - Sequential execution: some hours to some days
- UTS: standard benchmark

Experimental Validation

Application settings

- Parallel B&B: Taillard' Flowshop Instances (Ta20*20)
  - Permutational FSP: 20 jobs on 20 machines
  - Sequential execution: some hours to some days
- UTS: standard benchmark

Baseline algorithms

- H-MW: Hiearchical Adaptive MW (B&B-specific) [Bendjoudi et al., FGCS'12, IEEE TC'13]
- MW: Master-Worker (B&B-specific) [Mezmaz et al., IPDPS'07]
- RWS: (Distributed) Random work stealing [Dinan et al., SC'09]

Experimental Validation

Application settings

- Parallel B&B: Taillard' Flowshop Instances (Ta20*20)
  - Permutational FSP: 20 jobs on 20 machines
  - Sequential execution: some hours to some days
- UTS: standard benchmark

Baseline algorithms

- H-MW: Hiearchical Adaptive MW (B&B-specific) [Bendjoudi et al., FGCS'12, IEEE TC'13]
- MW: Master-Worker (B&B-specific) [Mezmaz et al., IPDPS'07]
- RWS: (Distributed) Random work stealing [Dinan et al., SC'09]

Grid5000 experiments

- 2 Clusters at Nancy site
  - Griffon: 736 cores of Intel 2.5 GHz
  - Graphene: 576 cores of Intel 2.6 Ghz

Our approach *vs.* H-MW

Average Scale (200 peers)

Speed-up w.r.t H-MW [FGCS'12]

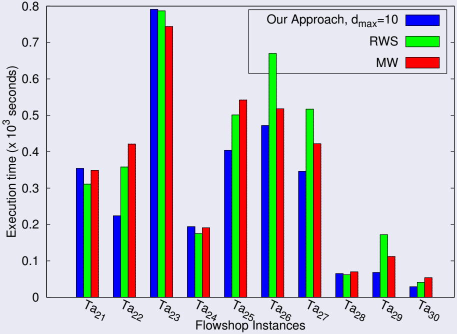
Flowshop$20 * 20$	Tree	Tree (asyn.)
Ta_{21}	**31.67**	**44.64**
Ta_{22}	**1.01**	**1.95**
Ta_{23}	0.65	0.98
Ta_{24}	**9.1**	**17.27**
Ta_{25}	**3.48**	**6.56**
Ta_{26}	**4.86**	**6.84**
Ta_{27}	0.85	**1.28**
Ta_{28}	**10.78**	**18.58**
Ta_{29}	0.98	**4.77**
Ta_{30}	**5.5**	**10.44**

- H-MW is adopting a BFS B&B-specific tree traversal strategy
- H-MW maps the B&B tree into the hierarchy to distribute work

Our approach *vs.* RWS *vs.* MW

Average Scale (200 peers)

Execution Time (200 peers)

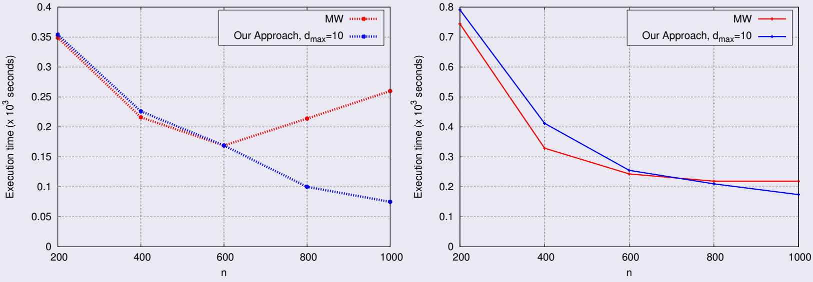


- Bottleneck in the MS centralized approach is negligible
- Both distributed and centralized schemes perform well

Our appraoch vs MW

Large Scale (1000 peers)

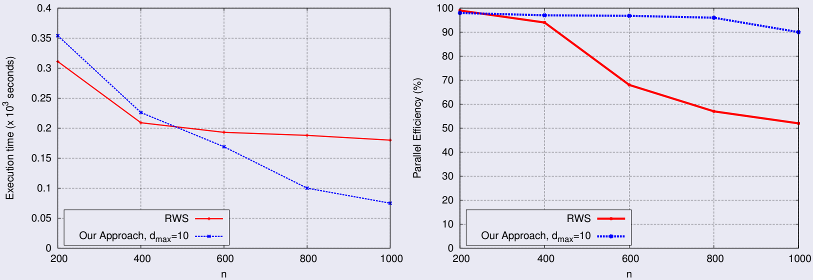
Scalability (up to 1000 peers)



(a) Ta21 (b) Ta23

- MW suffers from bottleneck when scaling the system
- The size of the B&B tree is not constant when scaling nodes

Scalability (up to 1000 peers)



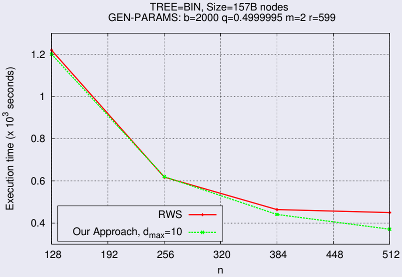
(c) Execution Time (d) Parallel Efficiency

Flowshop instance Ta21

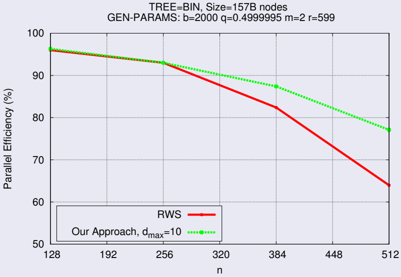
Our approach *vs.* RWS

Large Scale (B&B 1000 peers, UTS 512 peers)

Scalability (up to 1000 peers)



(e) Execution Time



(f) Parallel Efficiency

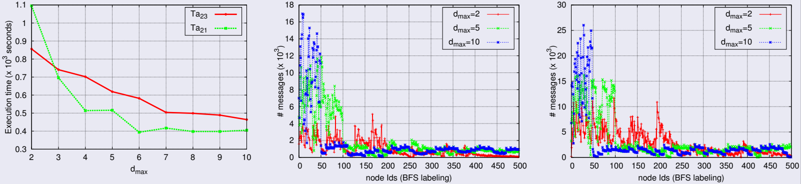
UTS instance

Discussion and Limitations

Discussion and Limitations

Overlay impact

- Small degree (large diameter) *vs.* Large degree (small diameter)



Flowshop Ta_{21} and Ta_{23} (500 peers)

Discussion and Limitations

Overlay impact

- Small degree (large diameter) *vs.* Large degree (small diameter)

Fault-tolerance

- Very large scale COPs cannot be solved in a single run

Discussion and Limitations

Overlay impact

- Small degree (large diameter) *vs.* Large degree (small diameter)

Fault-tolerance

- Very large scale COPs cannot be solved in a single run

Heterogeneity

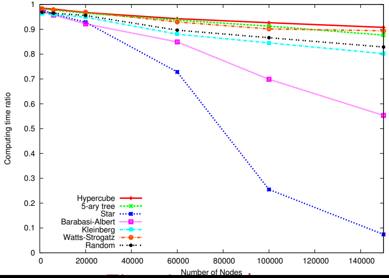
- Logical *vs.* physical
- Mapping of multi-* resources

Related results

Scalable overlays and Fault-tolerance

Peer-to-Peer inspired approach [done]

- Hypercube, Small world graphs, etc.
- 'Simulations' results (up to 2000 cores and 8 sites) on Grid5000



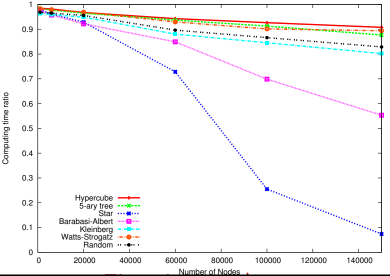
Scalable overlays and Fault-tolerance

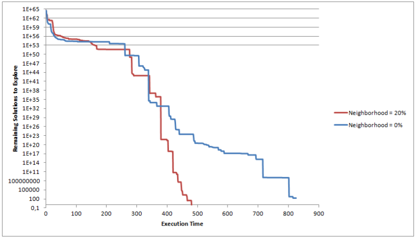
Peer-to-Peer inspired approach [done]

- Hypercube, Small world graphs, etc.
- 'Simulations' results (up to 2000 cores and 8 sites) on Grid5000

A hybrid fault-tolerant extension [paper in preparation]

- Centralized Checkpointing
- Distributed P2P work sharing

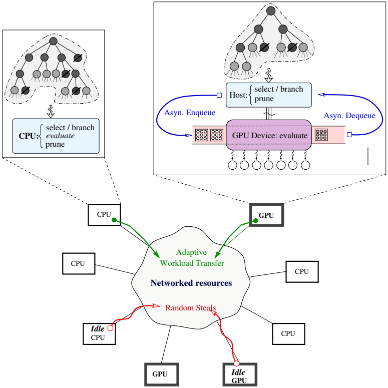




Random Failures (300 peers)

Heterogenous computing power / ability [done]

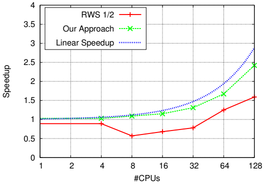
- CPU *vs.* GPU



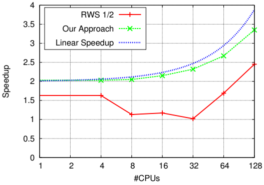
Heterogeneity

Heterogenous computing power / ability [done]

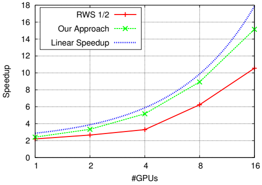
- CPU *vs.* GPU
- Near optimal parallel B&B with up to 20 GPU and 128 CPUs using 3 clusters of Grid5000



(g) #GPU = 1



(h) #GPU = 2



(i) #CPU = 128

Heterogeneity

Heterogenous computing power / ability [done]

- CPU *vs.* GPU
- Near optimal parallel B&B with up to 20 GPU and 128 CPUs using 3 clusters of Grid5000

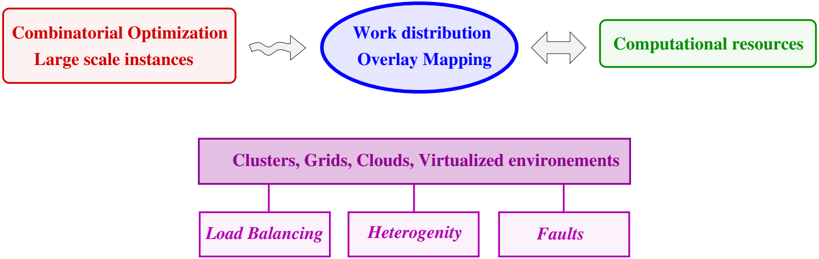
Heterogenous networks [Experiments in progress]

- Large scale distributed peers (e.g., latencies, throughput)
  - Overlay mapping using advanced graph structures?
  - Validation and performance assessment?
- Emulation using **Distem** on Grid5000

Conclusion

Next steps in COPs challenge

1. Solve very large scale problem instances
2. Heterogenous distributed, networked and virtualized resources



Thank You !

Questions ?