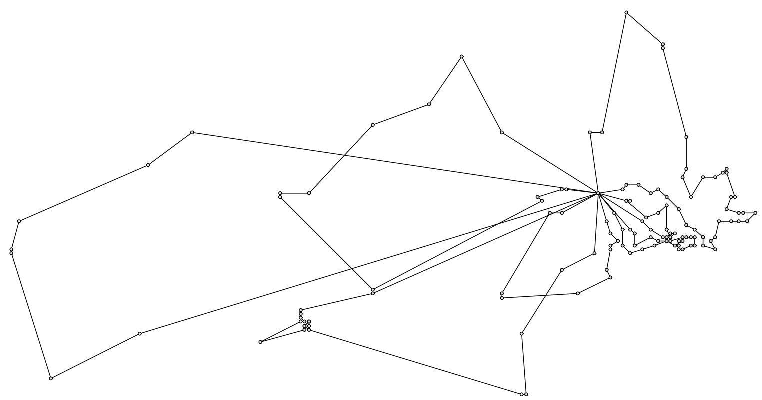
Parallel Solution of Vehicle Routing Problems

Ted Ralphs and Ali Pilatin

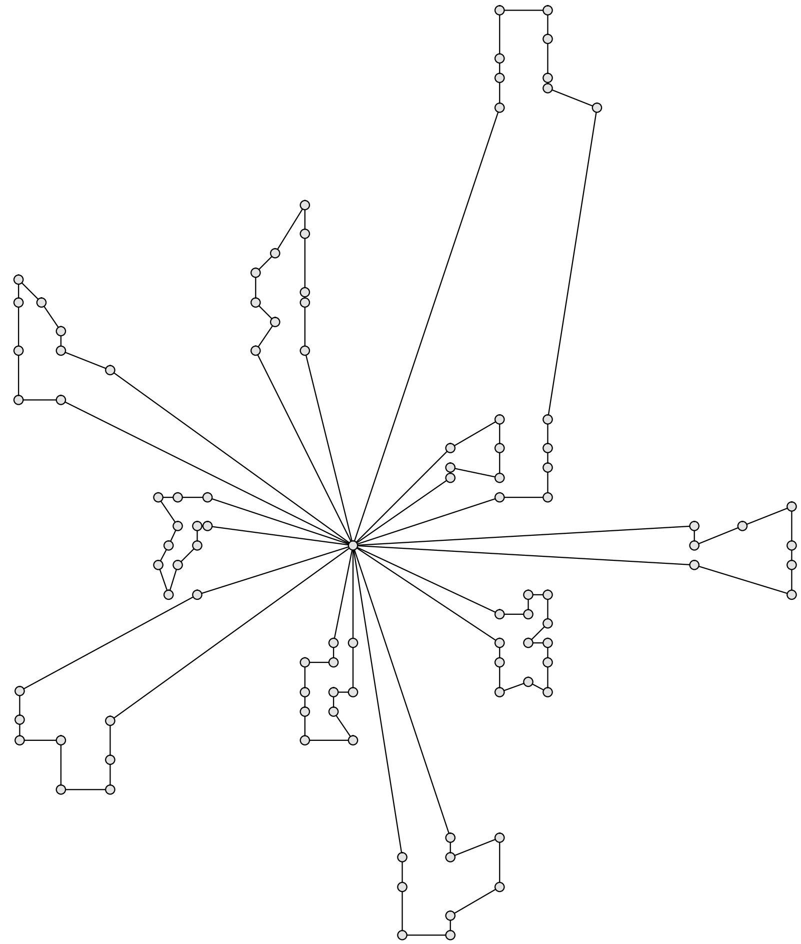
Industrial and Systems Engineering
Lehigh University



INFORMS Annual Meeting, November 16, 2005

Outline of Talk

- Introduction
- A Tale of Two Methodologies
- Performance Measures
- Algorithm Design
- Computational Results?
- Future Directions



The Project

- Goal: To develop optimization software for use in a commercial application for automated vehicle location and routing.
- The Holy Grail
  - Good solutions in "real time."
  - The ability to deal with real-time data "on the fly."
- Toolbox
  - Heuristic methods
    * Metaheuristics
    * Primal heuristics
  - "Exact" methods
    * Warm starting
    * Sensitivity analysis
    * Decomposition
  - Parallel/grid/on-demand computing
- Challenge: Create a hybrid technology that will use these tools in a synergistic manner.

The Setting

- Target users are small- and medium-sized delivery companies.
- We are interested in the usual variants of the Vehicle Routing Problem.
- Customer static input data
  - Location
  - Demand
  - Time window
  - Driver/vehicle restrictions
- Fleet static input data
  - Number of trucks
  - Truck capacities
  - Route length restrictions
  - Driver/vehicle descriptions
- Additional real-time data is available.
- The objective is to minimize total driving distance/time, subject to appropriate constraints.

A Tale of Two Methodologies

- "It was the best of solutions, it was the worst of solutions..."
- Heuristic methods produce *primal* information
  - Primal solutions
  - Upper bounds
  - Core variables
- Exact methods produce *dual* information.
  - Dual solutions
  - Lower bounds
  - Reduced costs
- The goal is to leverage this information in order to improve overall performance.
- What do we mean by "performance?"

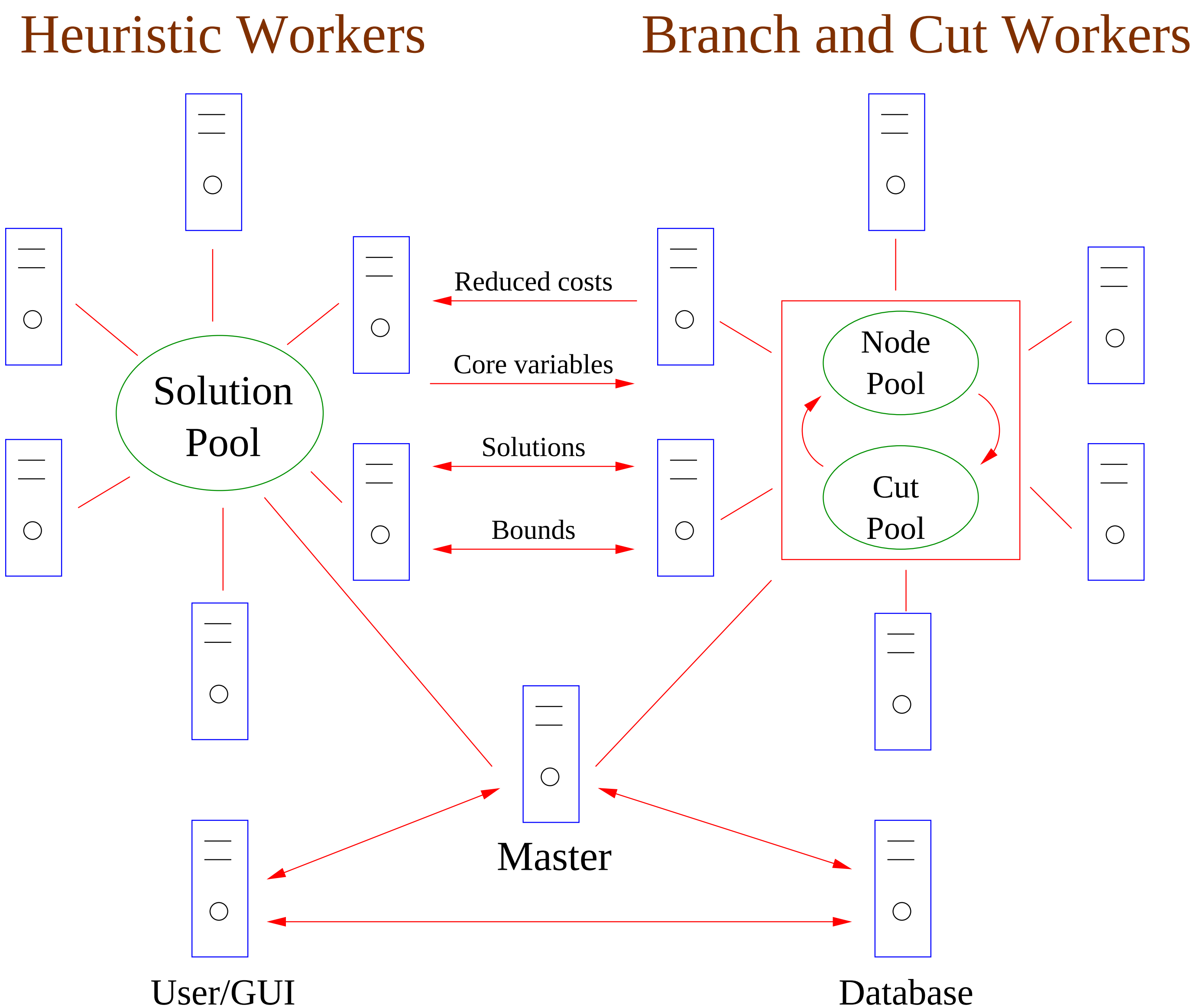
Performance Measures

- What is the goal anyway?
  - Solve a given problem instance as quickly as possible.
  - Find the best possible solution in a fixed amount of time.
  - Achieve the smallest possible gap in a fixed amount of time.
  - Some combination...
- Things to keep in mind.
  - The goal should be determined by the user.
  - The method should be tailored to the goal.
  - The goal may change as the solution procedure unfolds.
- Overall, we would like a flexible methodological framework that adjusts to the user's requirements.

A Bigger Hammer

- The easiest way to achieve real-time performance is to use a bigger hammer (parallel computng).
- Three phase parallel execution strategy
  - Phase I: Generate pool of initial solutions
  - Phase II: Improve pool with tabu search
  - Phase III: Close the gap (improve upper and lower bounds)
- Workers are allocated dynamically to perform one of three tasks
  - Construct solutions from scratch
  - Iteratively improve existing solutions
  - Process subproblems arising from branch and cut
- Workers can be migrated from one task to another dynamically.
- As with any parallel algorithm, the key is knowledge sharing.
- The goal is for branch and cut to work symbiotically with metaheuristic techniques.

Architecture Summary



Heuristics: Generating Initial Solutions

- During Phase I, all workers are dedicated to generating initial solutions.
- Initial generation is accomplished in several rounds.
  - Clustering: Generate an initial pool of solutions using a variety of simple clustering heuristics.
  - Routing: Improve initial routes for each of the clusters.
  - Exchange: Insert single customers on other routes
  - Routing: Run route improvement routines again.
  - Exchange: Exchange pairs of customers.
  - Routing: Run route improvement routines once more.
- In the initial clustering, each worker executes one or more heuristics and reports the results.
- The total number of heuristics to run is determined based on the size of the problem and the number of available processors.
- The steps are optional and the pool of solutions can be reduced after each step.

Heuristics: Initial Solutions for the CVRP

- Clustering heuristics
  - Sweep
  - Clarke and Wright
  - Savings Insertion
  - Near Cluster
  - Route First, Cluster Second
- Routing is done using standard techniques for TSP tour construction.
- The first round of exchange checks to see whether single customers can be inserted on other routes.
- The second round of exchange checks whether pairs of customers can be swapped.

Heuristics: Tabu Search

- During Phase II, all workers are dedicated to improving the most promising solutions with tabu search.
- This is done in parallel, with one or more solutions assigned to each available processor.
- Dynamic tabu tenures are used to handle diversification and intensification.
- Dives into the infeasible region are allowed with the optional penalized tabu approach.
- At each iteration, the neighborhood is (currently) defined by one of four randomly chosen moves.
  - Moving a node within a route.
  - Moving two nodes within a route.
  - Moving a node to another route.
  - Moving two nodes to other routes.

Exact Optimization: Branch and Cut

- The branch and cut algorithm is implemented using SYMPHONY.
- SYMPHONY is an open-source software package for solving and analyzing mixed-integer linear programs (MILPs).
- SYMPHONY can be used in three distinct modes.
  - Black box solver: Solve generic MILPs (command line or shell).
  - Callable library: Call SYMPHONY from a C/C++ code.
  - Framework: Develop a customized solver or callable library with callbacks
- SYMPHONY is part of the Computational Infrastructure for Operations Research (COIN-OR) libraries (`www.coin-or.org`).
- We have used SYMPHONY to implement a solver customized for the VRP.
- The solver is based on the standard two-index formulation and includes cut generation, custom branching rules, etc.
- If you want to know more about SYMPHONY, check out session WC.

Exact Optimization: Parallelizing Branch and Cut

- SYMPHONY can be used in a variety of parallel execution modes.
- SYMPHONY modules
  - Master
  - Tree Manager (TM)
  - Node Processor (NP)
  - Cut Generator (CG)
  - Cut Manager (CM)
- These modules can be combined to form executables that work in concert to perform branch and cut.
- We will use the following configuration.
  - One master consisting of the Master, TM, and CP modules.
  - Multiple workers consisting of the NP and CG modules.

Exact Optimization: Using the Solver

- The exact solver can be used effectively for optimizing individual routes.
  - Small TSPs can be quickly solved to optimality.
  - The routing step in initial solution generation is performed this way
- An exact TSP tour on all customer nodes can be used to initialize a route first, cluster second method.
- In Phase III, the exact solver can be used to:
  - Provide performance guarantees (lower bounds)
  - Generate improved primal solutions
  - Generate dual information to guide the heuristics (reduced costs).

Exact Optimization: Core Variables

- SYMPHONY allows the designation of a problem *core*.
  - A set of "important" constraints.
  - A set of variables "likely" to appear in an optimal solution.
- Designating a core can improve the efficiency of branch and cut.
- Optimizing over the core can also be an effective heuristic technique.
- SYMPHONY has a two-phase method in which the remaining variables are "repriced" in the root node.
- This technique can be used to remove variables from the problem before continuing the solution process.

Putting It All Together

- In Phase III, we can decide how many workers to allocate to each pool.
- We can continue to construct new solutions, improve existing ones, or focus on branch and cut.
- Knowledge sharing is the emphasis during this phase
- Knowledge passed from heuristics to branch and cut.
  - The core is composed of the union of the best heuristically generated tours.
  - The best solution found so far is used to initialize the upper bound.
- Knowledge passed from branch and cut to heuristics.
  - Solutions found during the search process become candidates for tabu search.
  - The indices of variables that have been fixed can also be passed to help guide execution of the heuristics.
  - Reduced costs and other dual information might be useful in guiding the heuristics.

Extensions: Heuristics Based on Branch and Cut

There are a number of ways in which "incomplete" branch and cut can be used as a heuristic or a solution improvement procedure.

- Switch to a search strategy emphasizing diving.
- Use a modified version of branch and cut to improve solutions
  - RINS-based local search.
  - Aggressive reduced-cost fixing.
- Since our goal is to find good solutions, default search parameters can be modified to emphasize finding feasible solutions.

Extensions: Real-time Changes to Problem Data

- We are currently exploring methodology for allowing dynamic changes to problem data.
- By maintaining appropriate solution information, we can adjust the solution procedure on the fly.
  - Pools of feasible solutions (primal)
  - Snapshots of the branch and cut tree (dual).
- For small changes, bound estimates can be obtained on the fly using sensitivity analysis techniques.
- For larger changes, warm starting techniques can allow the solution process to be restarted.
- Multicriteria and parametric optimization can be used to anticipate future changes.

Computational Setup

- We tested the solver on a variety of instances from the libraries maintained by Ralphs and Vigo.
- The solver was deployed on a Beowulf cluster with 60 1.8 GHz 64-bit AMD Opteron processors, each with 1G local memory.
- The communications protocol was PVM.
- Testing was done with numbers of processors varied from 1 to 32.
- Scaling strategy
  - first a scaling multiplier is calculated that is directly proportional to the number of processors, and inversely proportional to square of the number of vertices in the instance
  - Phase I: The number of clustering heuristics is scaled with the number of processors.
  - Phase II: Each processor received one solution to improve with tabu search.
  - Phase III: Exact solver was run with a time limit equal to twice the number of vertices.

Computational Results: Best Clustering Strategies

Heuristic	Average Percent over Best
TSP (FINI)	0.558991
TSP (FI)	1.17818
TSP (NI)	2.21181
Clarke and Wright	3.01447
Savings Insertion I	13.8535
Savings Insertion II	13.6061

Computational Results: Effect of Parallelism on Gap

Processors	Avg Final Gap without Using Core Variables
1	1.87
4	1.60
8	1.37
16	1.23
32	1.20

Computational Results: Scalability of SYMPHONY VRP Solver

Instance	Tree Size	Ramp Up	Ramp Down	Node Pack	Idle Node	Idle Index	Idle Diving	CPU sec	Wallclock	Eff
hk48 − n48 − k4	110	0.00	0.00	0.00	0.03	0.01	0.01	16.77	17.35	
att − n48 − k4	384	0.00	0.00	0.01	0.05	0.04	0.05	36.61	37.77	
E − n51 − k5	41	0.00	0.00	0.00	0.01	0.01	0.01	14.86	15.49	
A − n39 − k5	1307	0.00	0.00	0.04	0.23	0.18	0.15	242.33	251.04	
A − n39 − k6	328	0.00	0.00	0.01	0.07	0.04	0.03	31.95	32.66	
A − n45 − k6	631	0.00	0.00	0.02	0.13	0.08	0.07	175.12	181.64	
A − n46 − k7	43	0.00	0.00	0.00	0.01	0.01	0.00	15.03	15.71	
B − n34 − k5	795	0.00	0.00	0.01	0.10	0.08	0.09	29.99	31.08	
B − n43 − k6	288	0.00	0.00	0.01	0.05	0.04	0.03	38.24	39.67	
B − n45 − k5	133	0.00	0.00	0.00	0.02	0.02	0.02	15.69	16.21	
B − n51 − k7	367	0.00	0.00	0.01	0.12	0.05	0.05	53.50	55.37	
B − n64 − k9	152	0.00	0.00	0.00	0.03	0.02	0.02	44.00	46.12	
A − n53 − k7	3056	0.00	0.00	0.17	1.16	0.39	0.33	1295.06	1362.33	
A − n37 − k6	5873	0.00	0.00	0.21	1.36	0.73	0.67	832.46	857.40	
A − n44 − k6	5963	0.00	0.00	0.31	1.95	0.80	0.67	1677.44	1741.78	
B − n45 − k6	2039	0.00	0.00	0.07	0.46	0.28	0.23	379.23	394.49	
B − n57 − k7	3205	0.00	0.00	0.17	1.11	0.42	0.41	976.52	1036.09	
1 NP	24715	0.00	0.00	1.03	6.89	3.20	2.86	5874.80	6132.22	1.00
2 NP′s	24680	19.48	0.00	1.01	6.65	3.18	2.91	5832.25	3054.99	1.01
4 NP′s	23930	74.77	0.00	0.96	7.03	3.05	2.73	5620.94	1488.74	1.03
8 NP′s	24366	221.67	3.56	0.97	8.41	3.36	3.00	5701.00	774.99	0.99
16 NP′s	25668	572.93	11.83	0.98	11.52	3.75	3.51	6090.37	437.66	0.87
32 NP′s	25516	1416.55	60.98	3.02	113.66	121.30	110.45	5998.93	257.71	0.74

Computational Results: Early Lessons

- The exact optimization phase does seem to improve solutions in many cases, but the computational cost.is high.
- We need to further develop heuristic variants of branch and cut.
- Using exact TSP optimization seems to lead to significant improvement in initial solutions.
- For relatively small instances, optimizing over the core variables does not seem to help much.
- This may nto be the case for larger instances.
- There is much more work to be done.

Conclusions and Future Work

- This work is part of a much bigger ongoing effort to develop real-time solution procedures for IP.
- Our goal is to make IP a tactical decision-making tool.
- Our experience so far indicates that this will be a fruitful, but challenging area of research.
- Parallel and grid computing technologies are a big enabler.
- Heuristic solution techniques can be combined with exact solution techniques in useful ways to improve overall performance.