A Parallel Architecture for the Generalized Traveling Salesman Problem

Max Scharrenbroich
AMSC 663 Project Proposal

Advisor: Dr. Bruce L. Golden
R. H. Smith School of Business
Background and Introduction

• What is the Generalized Traveling Salesman Problem (GTSP)?
  – Variation of the well-known traveling salesman problem.
  – A set of nodes to be visited is partitioned into clusters.
  – Objective: Find a minimum-cost tour visiting exactly one node in each cluster.
  – Example on the following slides...
GTSP Example

- Start with a set of nodes or locations to visit.
• Partition the nodes into clusters.
GTSP Example (continued)

• Find the minimum tour visiting each cluster.
Applications

• The GTSP has many real-world applications in the field of routing:
  – Mailbox collection and stochastic vehicle routing.
  – Warehouse order picking with multiple stock locations.
  – Airport selection and routing for courier planes.
Mathematical Formulation

- The GTSP can be formulated as an Integer Linear Program (ILP).
- **Graph**: $G(V, E)$, where $V$ is a set of vertices partitioned into $m$ clusters $\{V_1, V_2, \ldots, V_m\}$ and $E$ is a set of edges connecting the vertices.
- **Distance Matrix**: $C$ is a distance matrix defined on $E$, where $c_e$ is the weight of edge $e$.
- **Decision Variables**: $x_e$ and $y_v$ are 0-1 decision variables representing the solution edges and vertices respectively.

\[
\begin{align*}
\text{min} & \quad \sum_{e \in E} c_e x_e \\
\text{subject to:} & \\
\sum_{v \in V_k} y_v & = 1 \quad k = 1, 2, \ldots, m \\
\sum_{e \in \delta(v)} x_e & = 2 y_v \quad \text{for } v \in V \\
\sum_{e \in \delta(S)} x_e & \geq 2 y_v \quad S \subset V^*, v \in V^* - S
\end{align*}
\]
Algorithms for the GTSP

• Like the TSP, the GTSP is NP-hard.
• There exist exact algorithms that rely on smart enumeration techniques:
  – Brand-and-cut (B&C) algorithm (M. Fischetti, 1997)
  – Provided exact solutions to reasonably sized GTSP problems (48 ≤ n ≤ 442 and 10 ≤ m ≤ 89).
  – For problems with larger than 90 clusters the run time of the B&C algorithm began approaching one day.
Algorithms for the GTSP (continued)

- Heuristic approaches to the GTSP:
  - Generalized Nearest Neighbor Heuristic (C.E. Noon, 1988)
  - Random-key Genetic Algorithm (L. Snyder and M. Daskin, 2006)
  - mrOX Genetic Algorithm (J. Silberholz and B.L. Golden, 2007)*
Overview of Genetic Algorithms (GA)

- Proposed in the 1970’s by Holland.
- Stochastic search technique commonly used to find approximate solutions to combinatorial optimization problems.
- Inspired by the process of natural selection and the theory of evolutionary biology.
- Simulate the evolution of a population of solutions.
Overview of GAs (continued)

• Components of Genetic Algorithms:
• Selection Operator:
  – Select the best solutions (chromosomes) for breeding.
• Crossover Operator:
  – Combine the pairs of selected solutions in some way to produce new solutions.
• Mutation Operator:
  – Randomly modify some solutions to preserve population diversity.
• Termination Criteria:
  – Terminate after a number of iterations (or period of time).
  – Terminate after a better solution is not found within a number of generations
Overview of mrOX Genetic Algorithm

• First, what is mrOX?
• The mrOX is the crossover operator at the heart of the mrOX GA.
• Modified rotational ordered crossover operator.
• Modification of the TSP ordered crossover (OX) proposed by (Davis, 1985).
• Results in a more “intelligent” crossover than the OX.
Example of the GTSP OX

- Chromosomes are represented by path-lists.

\[ \text{P1} \left[ 1^2, 3^5, 2^3, 4^8 \right] \]

\[ \text{P1}' \left[ 1^2 | 3^5, 2^3 | 4^8 \right] \rightarrow \left[ - | 3^5, 2^3 | - \right] \left\{ - \right\} \left[ - | 3^5, 2^3 | 1^{11} \right] \left\{ 1 \right\} \left[ 4^7 | 3^5, 2^3 | 1^{11} \right] \left\{ 1, 4 \right\} \]

\[ \text{P2} \left[ 2^4, 3^5, 4^7, 1^{11} \right] \]

\[ \text{P2}' \left[ 2^4 | 3^5, 4^7 | 1^{11} \right] \]

\[ 1^2 = (\text{cluster})^{(\text{node})} \]
MrOX

• Modify the inserted sub-path resulting from the OX operator and find the best one.

• rOX – rotational + OX:
  – Creates rotations and reversals of the inserted sub-path.
  – Example sub-tour: {1, 2, 3}
    • Rotations: { {1, 2, 3} {2, 3, 1} {3, 1, 2} }
    • Reversals: { {3, 2, 1} {1, 3, 2} {2, 1, 3} }

• MrOX – modified + rOX:
  – For each set of sub-paths generated in rOX create combinations of each node in the clusters at the end-points.
  – { 1^{A, B}, 3, 2^{D, E} }:
    • {1^A, 3, 2^D} {1^A, 3, 2^E} {1^B, 3, 2^D} {1^B, 3, 2^E}
The Serial mrOX GA

- The mrOX GA starts by first isolating a number of sub-populations for a several generations.
- Breeds new solutions using the mrOX crossover operator.
- Applies tour improvement heuristics like 2-opt and 1-swap on improved child solutions.
- Preserves diversity with a 5% chance of mutation.
- Terminates after the algorithm does not produce a better result in 150 generations.
Why Parallelize?

• Speedup
  – Provide higher quality solutions in less time.
• Increased Problem Size
  – Utilize more resources to attack larger problem instances.
• Robustness
  – Many serial heuristics require multiple input parameters that need to be tuned experimentally.
  – Each process can use a different set of parameters to avoid manual tuning.
  – Perform consistently on a range of problem instances.
• Cooperation
  – Use cooperative mechanisms to guide the search to more promising regions of the search space.
Cooperation Schemes

• **No Cooperation**
  – Provides a useful benchmark for testing other cooperation schemes.

• **Solution Warehouse***
  – Workers periodically send solution updates to a central repository.
  – The repository synchronizes the workers to a set of the best solutions found so far.

• **Inter-Worker Communication**
  – Cooperation is structured on a specific topology.
  – Worker processes may only cooperate with their neighbors.
  – Example: Ring Topology
Classification of Parallel Meta-heuristics

- Three classifications from Crainic and Toulouse (2003)
- Type 1: Low-Level Parallelism
  - Attempts to speed up processing within an iteration of a heuristic method.
- Type 2: Partitioning of Solution Space
  - Partitions the solution space into subsets to explore in parallel.
- Type 3: Concurrent Exploration*
  - Multiple concurrent explorations of the solution space.
Parallel Approach to the GTSP

• Run multiple instances of the mrOX GA in parallel.
• The proposed architecture supports a type 3 classification: multiple concurrent explorations of the solution space.
• Implement the solution warehouse method of cooperation to guide worker processes to more promising regions of the search space.
Method of Approach

1. Develop a general parallel architecture for hosting sequential heuristic algorithms.*
2. Extend the framework provided by the architecture to host the mrOX GA and the GTSP problem class.
3. Implement the solution warehouse method of cooperation.
Implementation

• Initial Development and Validation:
  – Multi-processor PC running Linux O/S.

• Final Validation and Testing:
  – UMD’s Deepthought Cluster, Linux O/S, up to 64 nodes with at least 2 processors.

• Language and Libraries:
  – C/C++
  – Message Passing Interface (MPI) Libraries
  – POSIX Threads Library
Database

• Based on a subset of TSP instances from the well-known TSPLib – a library of TSP instances.
• Use existing code for partitioning the nodes into clusters using method in (M. Fischetti, 1997).
• Use a set of larger instances tested in (Silberholz and Golden, 2007).
  – Number of nodes between 400 and 1084.
  – Number of clusters between 80 and 200.
  – The serial mrOX is already fast on small problem instances.
  – Don’t have optimal results for larger instances but there are published results for tests of the mrOX GA and S&D GA on these instances.
Validation

1. Validate the parallel architecture by implementing a simple test algorithm with several test-cases.

2. Validate the parallel implementation of the mrOX GA using a single worker process.

3. Validate the parallel implementation of the mrOX GA using more than one worker process.
Testing

- Test how the parallel implementation scales with the number of processors.
- Use results (i.e. solution costs) from runs of the serial mrOX GA as a stopping criterion for the parallel implementation.
- Measure the run times while using different numbers of processors.
- Test the efficacy of the cooperation scheme using the no-cooperation scheme as a benchmark.
- Time permitting, try a different cooperation scheme.
References