APLIKASI PARALEL *BRANCH AND BOUND* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM* MENGGUNAKAN PUSTAKA MPICH DAN GLPK

F.X. Arunanto, Arie Hintono
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Email: anto@if.its.ac.id

ABSTRAK

Vehicle Routing Problem (VRP) merupakan permasalahan optimasi dalam Riset Operasional, yaitu permasalahan melakukan pengantaran barang (komoditas) dari sebuah depot ke sejumlah pelanggan dengan lokasi dan jumlah permintaan barang yang diketahui. Karakteristik VRP yang diimplementasikan adalah VRP yang mempunyai kapasitas kendaraan sama, jumlah kendaraan tetap, batasan jarak dan meminimalkan jarak tempuh semua kendaraan sebagai tujuan. Untuk penyelesaian VRP, diimplementasikan algoritma paralel branch and bound yang dapat dijalankan oleh beberapa komputer untuk mendapatkan solusi eksak dengan waktu komputasi yang lebih cepat.

Langkah-langkah penyelesaian VRP dengan algoritma paralel branch and bound dimulai dengan tahap inisialisasi, yang menghasilkan subproblem yang berupa program linear pada search tree, kemudian dilakukan pembagian subproblem ke sejumlah komputer untuk dikerjakan masing-masing komputer. Setelah proses komputasi selesai, diperoleh solusi optimal beserta informasi rute terbaik yang ditemukan.

Uji coba dan evaluasi dilakukan dengan data yang dihasilkan secara acak, untuk jumlah pelanggan sebanyak 15-40, dengan menggunakan 1, 2, 4, dan 8 komputer. Dari hasil pengukuran waktu komputasi, didapat bahwa penambahan jumlah komputer pada proses komputasi dapat mempercepat waktu komputasi, bahkan dapat mencapai superlinear speedup terutama untuk jumlah pelanggan yang besar, akan tetapi pada suatu saat penambahan jumlah komputer malah memperlambat waktu komputasi, karena overhead waktu komunikasi yang semakin besar.

***Kata Kunci**: Vehicle Routing Problem, Optimization, Operational Research, Branch and Bound, Parallel Computing*.

1. PENDAHULUAN

Vehicle Routing Problem (VRP) merupakan persoalan mengenai sejumlah pelanggan dengan lokasi geografis diketahui yang memerlukan barang-barang yang harus dilayani dari sebuah depot dengan menggunakan sejumlah kendaraan yang memiliki kapasitas tertentu. VRP termasuk permasalahan optimasi pada bidang Riset Operasional. Dalam kehidupan, VRP berperan penting dalam bidang distribusi dan logistik.

Tujuan dari VRP adalah untuk menentukan rute pengantaran yang paling optimal, untuk setiap kendaraan, sehingga fungsi objektifnya terpenuhi.

Fungsi-fungsi objektif yang bisa dihasilkan dari VRP antara lain adalah sebagai berikut:

- Meminimalisasi jarak tempuh total.
- Meminimalisasi waktu tempuh total.
- Meminimalisasi jumlah kendaraan.

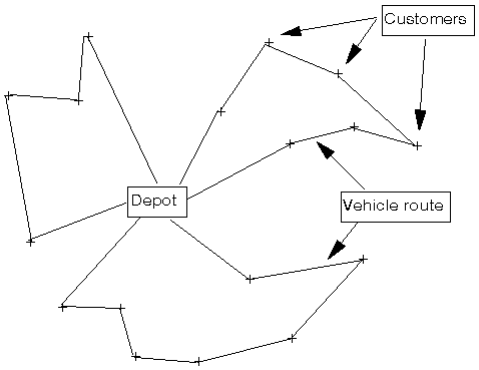
Batasan-batasan yang harus dipenuhi dalam penyelesaian VRP adalah sebagai berikut:

- Setiap permintaan barang dari pelanggan dapat terpenuhi.
- Setiap pelanggan dikunjungi hanya satu kali dan hanya oleh satu kendaraan saja.
- Setiap rute kendaraan bermula dan berakhir di depot.
- Jumlah total dari barang yang dapat diantar oleh sebuah kendaraan tidak lebih besar dari kapasitas kendaraan tersebut.

VRP secara konseptual adalah merupakan interseksi dari dua permasalahan yaitu:

- *Travelling Salesman Problem* (TSP)
  TSP merupakan permasalahan pencarian lintasan yang memberikan jarak minimum dari seorang pedagang yang berangkat dari suatu kota dan harus melalui semua kota tepat satu kali, sampai akhirnya kembali ke kota asal.
- *Bin Packing Problem* (BPP)
  BPP merupakan permasalahan memasukkan sejumlah item ke dalam sejumlah tempat penyimpanan sehingga total berat, volume, dan sebagainya tidak melebihi nilai maksimum.

Gambar 1 mengilustrasikan contoh VRP.



Gambar 1. Contoh VRP

Ada beberapa metode yang digunakan untuk menyelesaikan VRP berdasarkan [1] dan [2]. Metode tersebut dibagi dua bagian yaitu:

- Metode eksak
  Kelebihan metode ini adalah menghasilkan solusi yang eksak sedangkan kekurangannya adalah membutuhkan waktu yang lebih lama untuk proses komputasinya.
- Metode heuristik
  Kelebihan metode ini adalah biasanya menghasilkan solusi dengan waktu yang lebih cepat dibandingkan metode eksak, sedangkan kekurangannya adalah tidak menghasilkan solusi eksak tetapi konvergen mendekati ke suatu solusi.

Salah satu metode eksak yang bisa digunakan dalam penyelesaian VRP adalah dengan algoritma *branch and bound*. Akan tetapi, proses perhitungan dengan menggunakan algoritma *branch and bound* ini akan memakan waktu yang panjang terutama untuk jumlah pelanggan/kota yang semakin banyak. Oleh karena itu diperlukan cara untuk mempercepat waktu komputasi.

Salah satu cara yang dilakukan untuk mengatasi lamanya waktu komputasi adalah dengan menggunakan sistem paralel, yaitu melakukan komputasi dengan menggunakan lebih dari satu komputer secara bersamaan, yang dapat mengatasi kelemahan dari sistem sekuensial.

Salah satu sistem paralel yang dapat digunakan adalah paradigma *Message Passing*. Pada sistem ini komputasi dikerjakan oleh sejumlah prosesor dimana saat proses komputasi tersebut, prosesor-prosesor tersebut berkomunikasi satu sama lain dengan cara mengirimkan *message*. Salah satu pustaka yang merupakan penerapan dari paradigma *Message Passing* adalah *Message Passing Interface* (MPI) [3].

2. FORMULA VRP

Formula VRP berdasarkan [4] dan [5] dapat diuraikan seperti berikut ini. VRP dapat direpresentasikan dengan graph sebagai berikut:
Graph G = (N, E) dimana:

- N = {1, 2, 3, …, n}
- 1 = lokasi depot, dan δ = {2, 3, …, n} adalah lokasi pelanggan
- E = {(i, j) ; i, j ε N, i < j }
- Q_j = permintaan dari pelanggan j
- V = jumlah kendaraan
- W = kapasitas dari masing-masing kendaraan
- C_{ij} = jarak dari lokasi i dan j
- T = jarak tempuh maksimum kendaraan
- L_i = jarak dari *shortest* (n,i) *path* pada G

Fungsi Objektif VRP yang digunakan adalah meminimalisasi jarak tempuh total untuk semua kendaraan, dan dapat diekspresikan dalam formula matematika berikut:

$$Minimalkan\ f = \sum_{i<j} C_{ij}X_{ij} \quad (1)$$

X_{ij} mempunyai beberapa kemungkinan nilai, yaitu: 1 jika kendaraan digunakan pada perjalanan tunggal antara i dan j (i, j ε N}, 2 jika i = 1 dan (1, j, 1) adalah sebuah rute, serta 0 untuk selain kedua kondisi sebelumnya.

Persyaratan untuk mendefinisikan X_{ij} adalah:

$$i < j \quad (2)$$

$$Q_i + Q_j \le W \quad (3)$$

$$L_i + L_j + C_{ij} \le T \quad (4)$$

dengan mempunyai beberapa konstrain yang harus dipenuhi antara lain:

C1. Konstrain di depot, yaitu konstrain yang menjamin digunakannya V kendaraan.

$$\sum_{j\in\delta} X_{1j} = 2V \quad (5)$$

C2. Konstrain pada masing-masing pelanggan, yaitu konstrain yang menjamin setiap pelanggan hanya dikunjungi sekali.
untuk setiap $k \in \delta$,

$$\sum_{i<k} X_{ik} + \sum_{j>k} X_{kj} = 2 \quad (6)$$

C3. Konstrain integer pada variabel.

$$Xij = \begin{cases} 0,1, \text{atau } 2 & \text{untuk } i = 1 \\ 0,1 & \text{selain itu} \end{cases} \quad (7)$$

C4. V adalah integer positif.

C5. Konstrain yang dapat menghilangkan *cycle* untuk *subtour* yang tidak melibatkan depot, *subtour* yang

melanggar batasan kapasitas, atau *subtour* yang melanggar batasan jarak maksimum kendaraan.

Untuk setiap $S \subseteq \delta, |S| \geq 2,$

$$\sum_{\substack{i,j \in S \\ i<j}} X_{ij} \leq |S| - \ell(s) \tag{8}$$

dimana $\ell(s)$ untuk VRP dengan batasan kapasitas dan jarak, dihitung sebagai berikut:

$$\max \left\{ \left\lceil \frac{1}{w} \sum_{j \in S} Q_j \right\rceil, \left\lceil \frac{1}{T} \sum_{\substack{i,j \in S \cup \{n\} \\ i<j}} C_{ij} X_{ij} \right\rceil \right\} \tag{9}$$

3. ALGORITMA SEKUENSIAL BRANCH AND BOUND

Langkah-langkah penyelesaian VRP dengan algoritma sekuensial branch and bound dapat dilihat pada diagram alir pada Gambar 2.

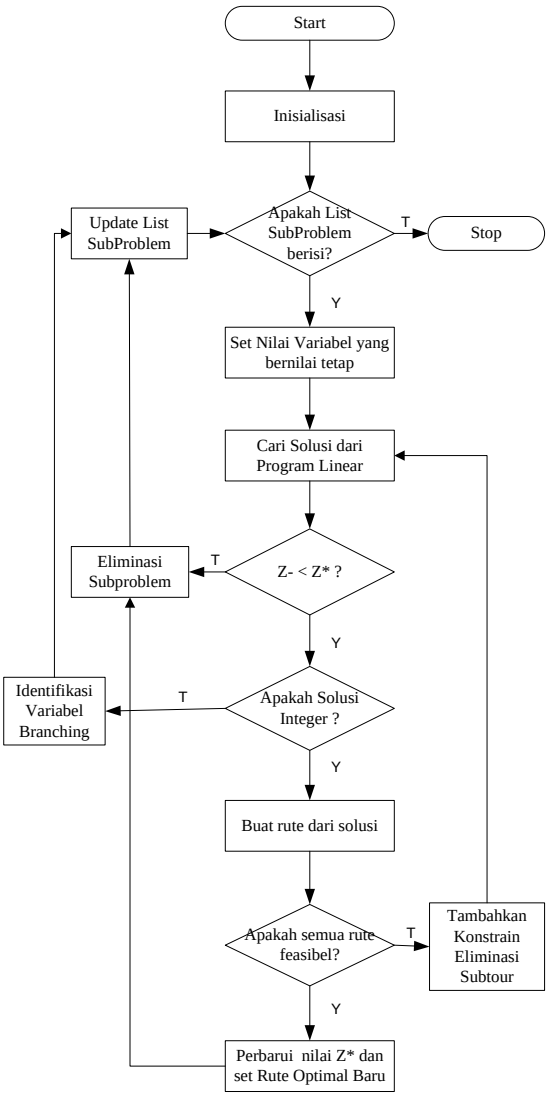
Pada gambar 2, Z* merepresentasikan nilai fungsi objektif terbaik untuk solusi feasibel yang diperoleh sampai saat ini dan Z- merepresentasikan nilai fungsi objektif (*lower bound*) untuk program linear dalam subproblem saat ini.

Beberapa hal penting pada algoritma sekuensial *branch and bound* untuk penyelesaian VRP ini antara lain adalah:

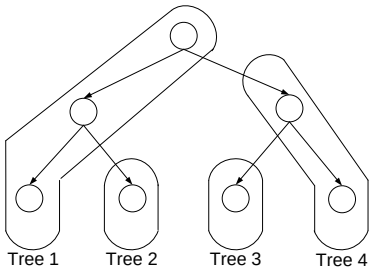
- Subproblem yang dihasilkan pada masing-masing node pada search tree adalah program linear yang didefinisikan oleh konstrain yang dihasilkan oleh batasan kapasitas kendaraan pada solusi node orang tuanya. Lebih lanjut, ketika melakukan *backtracking*, batasan kapasitas kendaraan tetap digunakan.
- Untuk memilih variabel percabangan, sebuah variabel yang bernilai fraksional yang paling mendekati 0 atau 1 dipilih. Percabangan pertama dibuat pada arah yang berlawanan.
- Tidak semua kombinasi yang mungkin harus diproses, sebagian node pada *search tree* dapat dipotong karena mereka mempunyai fungsi objektif yang lebih besar daripada Z*.
- Perhitungan antara node dalam *search tree* tidak tergantung satu dengan yang lainnya.

4. ALGORITMA PARALEL BRANCH AND BOUND

Pada algoritma paralel *branch and bound*, subproblem-subproblem pada *search tree* dibagi kepada sejumlah prosesor, seperti yang dapat dilihat pada Gambar 3.



Gambar 2. Diagram Alir Proses Penyelesaian VRP dengan Algoritma Sekuensial *Branch and Bound*



Gambar 3. Pemetaan *search tree* ke sejumlah prosesor

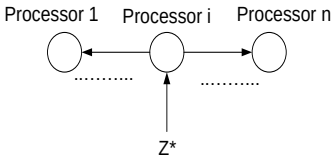
Pada algoritma paralel *branch and bound*, masing-masing subproblem/node anak dari *search tree* perlu mewarisi program linear dari orang tuanya. Oleh karena itu, merupakan hal yang ideal untuk melakukan pemetaan sebuah subproblem kepada

masing-masing prosesor sehingga subproblem yang berbeda dapat dicari secara independen tidak bergantung satu dengan yang lainnya. *Search tree* yang ada di sini merupakan *binary search tree* dan oleh karena itu ada 2k subproblem pada level ke-k dari *search tree*.

Algoritma Paralel yang diimplementasikan di sini menggunakan cara SPMD (*Single Program Multiple Data*). Dengan kata lain, *master* dan *slave* mengeksekusi program yang sama (mengeksplorasi subproblem-subproblem mereka sendiri) pada kumpulan data yang berbeda.

Jika salah satu prosesor menemukan *upper bound* yang lebih baik (Z*) dalam proses pencariannya, maka dia perlu mengirimkan nilai Z* ini ke semua prosesor lain untuk memperbarui nilai Z* pada semua prosesor yang ada, seperti yang dapat dilihat pada Gambar 4.

Ketika semua prosesor sudah menyelesaikan proses pencarian mereka, maka prosesor yang memegang solusi yang paling optimal akan mengirimkan informasi rute dan nilai fungsi objektif kepada master, yang merupakan solusi optimal yang ditemukan.



Gambar 4. Memperbarui nilai Z* ke semua prosesor

5. IMPLEMENTASI

Bahasa pemrograman yang digunakan untuk membuat perangkat lunak ini adalah C++, sedangkan antarmukanya menggunakan Qt versi 3.1 pada sistem operasi Linux, toolkit yang mendukung pengembangan aplikasi C++ yang multiplatform dengan dukungan antarmuka grafis.

Ada dua *library* utama yang digunakan untuk membantu pembuatan perangkat lunak ini, yaitu: MPICH dan GLPK (GNU Linear Programming Kit).

MPICH adalah *library* yang merupakan implementasi portabel dari standar *Message Passing Interface* (MPI) dan bersifat *open source*. Dalam pembuatan perangkat lunak ini, MPICH digunakan untuk mendukung proses paralel, yaitu untuk pertukaran pesan antara prosesor satu dengan prosesor yang lain. Versi yang digunakan dalam pembuatan perangkat lunak ini adalah MPICH versi 1.2.5.

GLPK merupakan *library* yang diperuntukkan untuk menyelesaikan permasalahan pemrograman linear atau *linear programming* (LP) dalam skala besar, pemrograman linear integer campuran atau *mixed integer linear programming* (MIP), dan permasalahan yang berhubungan [6].

Dalam pembuatan perangkat lunak ini, GLPK digunakan untuk menyelesaikan permasalahan pemrograman linear (*LP Solver*) yang terdapat pada node-node dalam tree yang terbentuk, dengan menggunakan metode simpleks. Versi yang digunakan dalam pembuatan perangkat lunak ini adalah GLPK versi 4.1.

Implementasi proses sekuensial untuk penyelesaian VRP dengan algoritma *branch and bound*, adalah seperti ditunjukkan pada Gambar 5.

```
Inisialisasi();
while (jumlist>0) {
  is_solve = solve_subproblem();
  if (is_solve == 1) {
    if(VRP.is_zcurr_less_than_zopt()==1) {
      is_int = VRP.is_solution_integer();
      if(is_int==1) {
        VRP.buat_subtour();

  is_feasible=VRP.is_tour_feasible();
        if (is_feasible == 1) {
          VRP.set_optimal_tour();
          del_branch();
        }
        else{//Ada subtour tak feasibel
          add_elimination_constraint();
        }
      }
      else {    //Solution tidak integer
        branching();
      }
    }
    else { //Z current > Z optimal,
           //tidak perlu diteruskan
      del_branch();
    }
  }
  else {//Program Linear tak feasibel
    del_branch();
  }
}
```

Gambar 5. *Pseudocode* penyelesaian VRP dengan algoritma *Branch and Bound*

Fungsi-fungsi untuk proses pengiriman pesan secara paralel yang diperlukan antara lain:

- Fungsi untuk mengirim data awal VRP ke semua prosesor dari prosesor pertama.
- Fungsi untuk menerima data awal VRP yang dikirim oleh prosesor pertama.
- Fungsi untuk mengirimkan nilai Z* baru yang ditemukan pada satu prosesor, agar semua prosesor dapat memperbarui nilai Z* nya.
- Fungsi untuk menerima nilai Z optimal baru yang dikirim oleh prosesor lain.
- Fungsi untuk mengirimkan subproblem ke prosesor lain.

- Fungsi untuk menerima subproblem yang dikirim dari prosesor lain.

6. HASIL UJI COBA

Uji coba perangkat lunak yang dihasilkan dilakukan di Laboratorium Komputing Jurusan Teknik Informatika ITS Surabaya, dengan menggunakan sejumlah komputer. Maksimal komputer yang digunakan untuk uji coba ini adalah 8 komputer dengan spesifikasi komputer yang sama antara komputer yang satu dengan yang lain yaitu komputer dengan spesifikasi prosesor Pentium II 450 MHz, dengan memori 320MB.

Untuk proses pembangkitan data dilakukan secara acak pada perangkat lunak ini, menggunakan beberapa kriteria untuk menghasilkan data yang baik sebagai berikut.

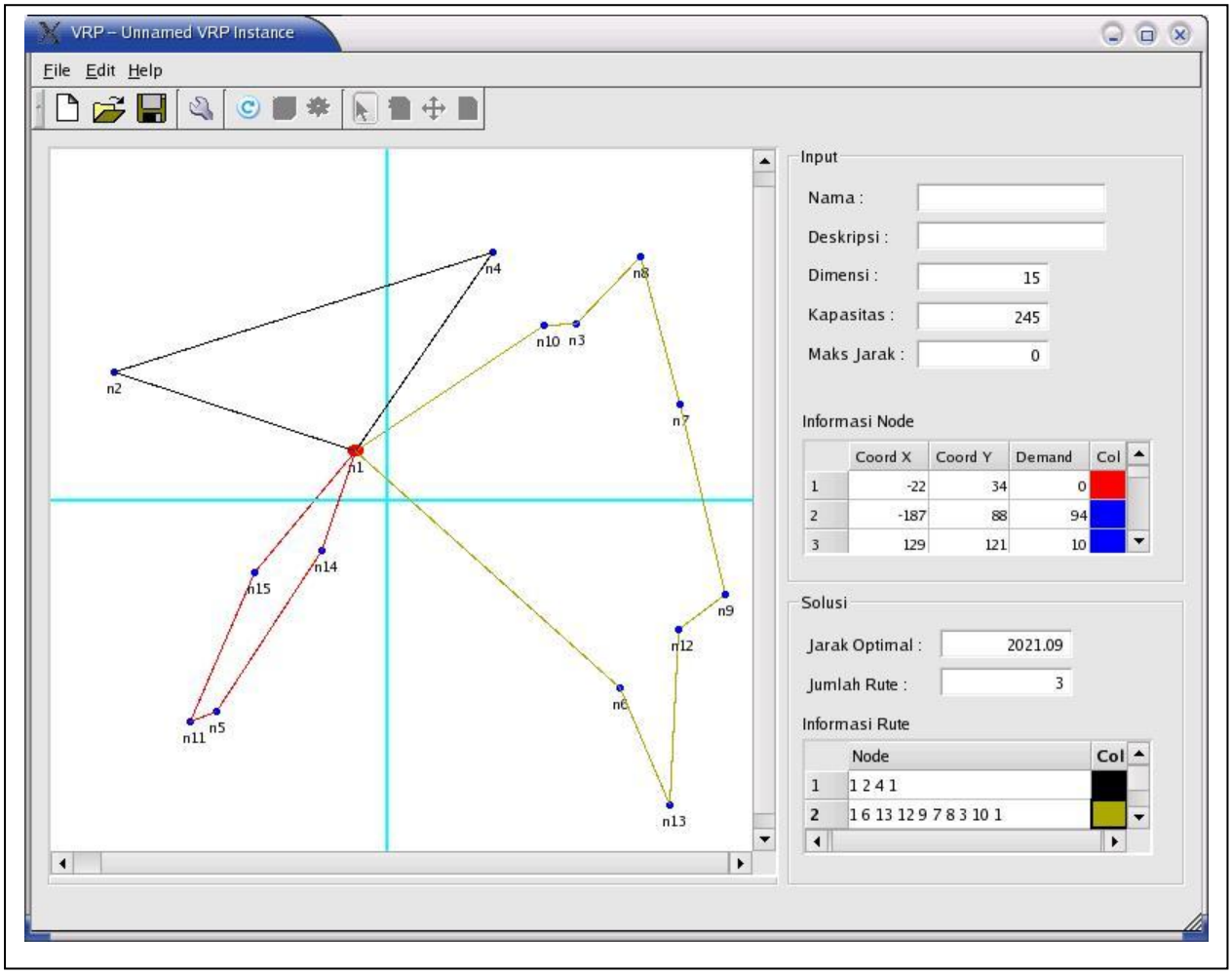
- Koordinat lokasi pelanggan bernilai integer dan dibangkitkan secara acak dengan batas koodinat 1000.
- Permintaan pelanggan dibangkitkan secara acak dengan nilai antara 1 sampai 100.
- Kapasitas kendaraan ditentukan dengan $W = (1-\alpha) \max Qi + \alpha \sum Qi$
  α adalah parameter yang akan menentukan nilai kapasitas kendaraan yang bernilai antara 0-1, dimana nilai-nilai yang umum digunakan untuk α adalah 0, 0,33, 0,67, dan 1. Untuk uji coba ini digunakan α bernilai 0,33 dan 0,67.

Untuk uji coba antarmuka, dilakukan uji coba dengan parameter-parameter masukan dari VRP yang digunakan adalah nilai $\alpha = 0.33$, jumlah node = 15, dan kapasitas kendaraan = 245, dengan data-data node yang dihasilkan secara acak.

Terdapat 3 rute yang dihasilkan, yaitu:

- Rute 1: 1 2 4 1
- Rute 2: 1 6 13 12 9 7 8 3 10 1
- Rute 3: 1 14 5 11 15 1

dengan nilai jarak optimal yang ditemukan adalah 2021,09. Hasil dari tampilan untuk keluaran proses komputasi dapat dilihat pada Gambar 6.



Gambar 6. Keluaran untuk uji coba antarmuka

Tabel 1. Waktu komputasi, waktu komunikasi, dan waktu komputasi total

α	∑ Node	Waktu Komputasi (detik)				Waktu Komunikasi (detik)				Waktu Komputasi Total (detik)			
		1 proc	2 proc	4 proc	8 proc	1 proc	2 proc	4 proc	8 proc	1 proc	2 proc	4 proc	8 proc
0.33	15	10.554	0.961	0.778	0.204	-	3.165	0.701	2.299	10.554	4.126	1.479	2.503
	20	36.302	5.403	2.612	1.883	-	5.485	6.453	10.849	36.302	10.888	9.065	12.732
	25	71.491	7.697	6.101	2.459	-	40.936	13.054	5.943	71.491	48.633	19.155	8.402
0.67	20	7.695	1.708	1.956	0.856	-	1.667	4.069	6.187	7.695	3.375	6.025	7.043
	30	29.123	10.876	3.105	1.518	-	4.628	5.951	9.046	29.123	15.504	9.056	10.564
	40	338.358	175.318	36.167	28.169	-	102.839	55.575	86.861	338.358	278.157	91.742	115.030

Tabel 2. *Speedup* dan efisiensi proses paralel

α	∑ Node	SpeedUp			Efisiensi		
		2 proc	4 proc	8 proc	2 proc	4 proc	8 proc
0.33	15	2.558	7.136	4.217	1.279	1.784	0.527
	20	3.334	4.005	2.851	1.667	1.001	0.356
	25	1.470	3.732	8.509	0.735	0.933	1.064
0.67	20	2.280	1.277	1.093	1.140	0.319	0.137
	30	1.878	3.216	2.757	0.939	0.804	0.345
	40	1.216	3.688	2.941	0.608	0.922	0.368

Uji coba proses sekuensial maupun proses paralel dengan 2, 4, dan 8 prosesor dilakukan dengan nilai α = 0,33 untuk jumlah node 15, 20, dan 25, dan nilai α = 0,67 untuk jumlah node 20, 30, dan 40, hasilnya dapat dilihat pada Tabel 1.

Selanjutnya diukur *speedup* dan efisiensi dari program paralel yang dapat dilihat pada Tabel 2, dengan rumus sebagai berikut:

$$\text{SpeedUp} = \frac{\text{Waktu eksekusi untuk 1 prosesor}}{\text{Waktu eksekusi untuk n prosesor}} \quad (10)$$

$$\text{Efisiensi} = \frac{\text{SpeedUp}}{\text{Jumlah Prosesor}} \quad (11)$$

Dari Tabel 2 terdapat hasil dimana speedup > jumlah prosesor atau efisiensi > 1, contohnya untuk permasalahan dengan nilai α=0.33 dan jumlah Node=15 yang memenuhi adalah untuk 2 prosesor dan 4 prosesor. Kondisi ini dikatakan superlinear speedup.

Faktor yang memberikan kontribusi terhadap kondisi superlinear speedup ini adalah dengan semakin banyak proses yang melakukan pencarian Z* maka nilai Z* yang lebih baik akan ditemukan lebih cepat sehingga banyak subtree yang dapat dieliminasi sebelum diproses, sehingga dapat menghemat waktu komputasi dan dapat mengatasi overhead waktu komunikasi.

Penambahan jumlah prosesor pada proses paralel dapat mempercepat waktu komputasi bahkan dapat mencapai kondisi superlinear speedup seperti dijelaskan sebelumnya, terutama untuk jumlah pelanggan yang besar. Akan tetapi pada suatu saat penambahan jumlah prosesor untuk proses paralel tidak akan mempercepat proses komputasi lagi (terdapat titik jenuh penambahan prosesor) karena overhead waktu komunikasi yang lebih besar dibandingkan keuntungan dari lebih cepatnya penemuan Z* yang lebih baik.

7. SIMPULAN DAN SARAN

Dari hasil uji coba dan evaluasi perangkat lunak, dapat diambil kesimpulan sebagai berikut:

- Faktor-faktor yang mempengaruhi kecepatan dalam mendapatkan solusi untuk penyelesaian VRP secara umum adalah:
  1. jumlah pelanggan, dimana semakin banyak pelanggan semakin lambat prosesnya karena semakin banyak variabel (*edge*) yang pada program linear untuk setiap subproblem.
  2. nilai kapasitas dan jarak tempuh maksimum kendaraan dibandingkan dengan permintaan dan jarak pelanggan, semakin kecil nilainya maka prosesnya semakin lambat karena semakin banyak rute yang dihasilkan.
- Penambahan jumlah prosesor pada proses paralel dapat mempercepat waktu komputasi terutama untuk jumlah pelanggan yang besar bahkan dapat mencapai *superlinear speedup*, karena semakin banyak proses yang melakukan pencarian Z* maka nilai Z* yang baik akan ditemukan lebih cepat sehingga banyak *subtree* yang dapat dieliminasi sebelum diproses, sehingga menghemat waktu komputasi dan mengatasi *overhead* waktu komunikasi.
- Pada suatu saat penambahan jumlah prosesor untuk proses paralel akan memperlambat proses

komputasi karena *overhead* waktu komunikasi lebih besar dibanding keuntungan lebih cepatnya penemuan Z* yang lebih baik.

Berikut ini adalah saran-saran untuk pengembangan lebih lanjut:

- Menggunakan metode-metode lain selain algoritma *branch and bound* dalam penyelesaian VRP seperti metode *tabu search*, metode *simulated annealing*, algoritma genetik, dan sebagainya untuk mengetahui bagaimana pengaruhnya terhadap solusi yang ditemukan dan waktu komputasinya.
- Jenis VRP yang dibahas di sini adalah jenis *Capacitated* VRP (CVRP) dengan konstrain kapasitas dan konstrain jarak, terdapat jenis-jenis VRP lain yang dapat dicari penyelesaiannya seperti VRP dengan *time windows* (VRPTW), VRP dengan banyak depot (MDVRP), *Split Delivery* VRP (SDVRP), VRP dengan *pickup and delivering* (VRPPD), *Periodic* VRP (PVRP) dan lainnya.
- Algoritma *branch and bound* untuk penyelesaian VRP yang dikembangkan dalam perangkat lunak ini dapat dioptimasi lebih lanjut untuk menghasilkan solusi yang lebih cepat dengan lebih meningkatkan *lower bound* awal pada permasalahan, antara lain dengan menggunakan algoritma *branch and cut* yang merupakan pengembangan dari algoritma *branch and bound.*
- Untuk pemrosesan secara paralel, dapat digunakan *load balancing* untuk menyeimbangkan kerja antara prosesor yang satu dengan yang lain sehingga dapat mempercepat waktu komputasi.

8. DAFTAR PUSTAKA

[1] Laporte, G; Osman, H. "Routing Problems: A Bibliography", **Annals of Operations Research** 1995; 61:227-262.

[2] Laporte, G; Gendreau, M; Potvin, J. Y. "Classical and Modern Heuristics for the Vehicle Routing Problem", **International Transactions in Operational Research** 2000, 7:285-300.

[3] Snir, M; Otto, S. W; Huss-Lederman, S; Walker, D. W; Dongarra, J. **MPI-The Complete Reference**, MIT Press; 1998.

[4] Achuthan, N. R; Caccetta, L; Hill, S. P. "A New Subtour Elimination Constraint for the Vehicle Routing Problem", **European Journal of Operational Research** 1996; 91:573-586.

[5] Lau, K. K; Kumar, M.J. "Parallel Implementation of Branch and Bound Algorithm for Solving Vehicle Routing Problem on NOWs", **Proceedings of the Third International Symposium on Parallel Architectures, Algorithms, and Networks**, 1997. (I-SPAN '97), 1997, 247-253.

[6] Makhorin, A. **GLPK Reference Manual**, Department for Applied Informatics, Moscow Aviation Institute; 2003.