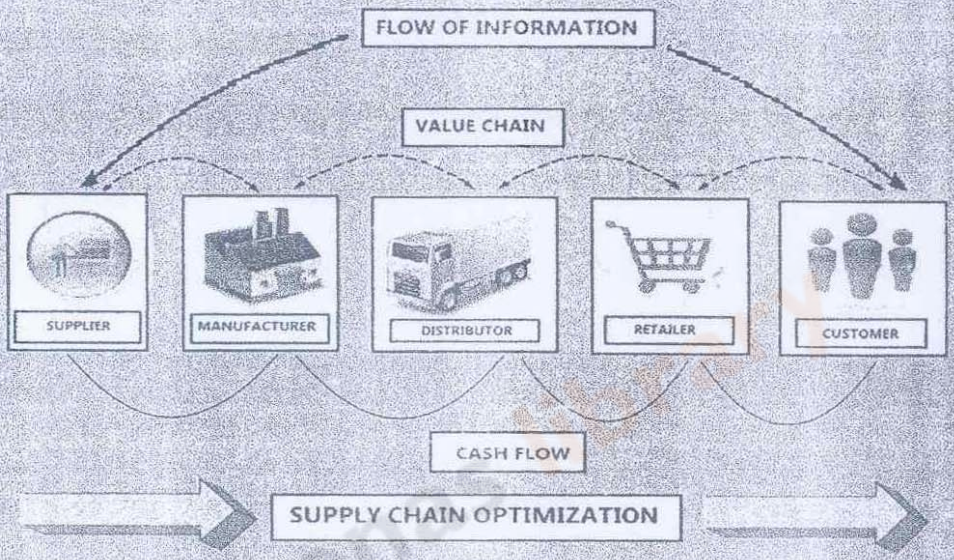
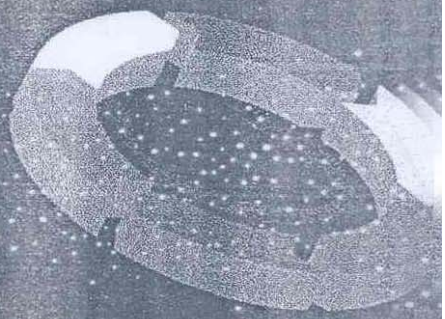
# Integrated Logistics and Informatin System for Developing Supply Chain Management in Logistic Industries




# PROCEEDING

## SEMINAR NASIONAL SUPPLY CHAIN DAN SISTEM INFORMASI (SEMASSI)

Volume No.1



MASYARAKAT LOGISTIK INDONESIA

Cabang Tanjung Priok

# Aplikasi Variable Neighborhood Descent Metaheuristik untuk Vehicle Routing Problem

Liane Okdinawati[1], Arif Imran[2]
[1]Jurusan Logistik Bisnis, Politeknik Pos Indonesia
[2]Jurusan Teknik Industri Institut Teknologi Nasional Jl.PHH Mustopha 23 Bandung, 40124

***Abstrak***

*Vehicle Routing Problem (VRP) diselesaikan dengan mengaplikasikan variable neighborhood descent (VND) metaheuristik. Solusi inisial diperoleh dengan menggunakan algoritma Dijkstra berdasarkan cost network yang dibentuk oleh algoritma Sweep dan 2-opt. Algoritma variable neighborhood descent yang diusulkan pada penelitian ini menggunakan beberapa neighborhood (local search ) untuk mendapatkan solusi. Sebagai tambahan pada algoritma juga diaplikasikan prosedur diversifikasi yang membantu proses pencarian solusi jika solusi yang didapatkan oleh local search- local search yang digunakan tidak dapat diperbaiki lagi. Algoritma usulan diuji dengan data set yang terdapat pada literatur.*

***Kata kunci***: metaheuristik, *local search*, *routing*, *variable neighborhood*.

## 1. Pendahuluan

Vehicle Routing Problem (VRP) mempunyai peranan penting dalam manajemen distribusi dan merupakan salah satu masalah yang dipelajari secara luas dalam optimisasi kombinatorik. VRP adalah masalah dimana sejumlah konsumen harus dilayani oleh sejumlah kendaraan yang sejenis yang berasal dari satu depot. Setiap konsumen dikunjungi sekali, kapasitas maksimum kendaraan dan panjang rute maksimum kendaraan tidak boleh dilanggar. Tujuan yang akan dicapai dengan dengan menyelesaikan VRP adalah mendapatkan satu set rute yang memenuhi persyaratan-persyaratan di atas dengan biaya yang terkecil.

Dua metoda untuk menyelesaikan VRP, metoda eksak dan metode heuristik. Untuk masalah yang kecil ( $n$<30), metoda eksak seperti *Integer Linear Programming* dapat memberikan soiusi dengan waktu komputasi yang cepat. Metoda eksak menjadi tidak efektif (memerlukan waktu komputasi yang lama) untuk masalah yang besar. Peningkatan waktu komputasi ini terjadi karena VRP merupakan masalah *NP-hard* (Lenstra dan Rinnooy Kan, [13]).

Terdapat banyak publikasi yang membahas cara penyelesaian VRP. Sebagai contoh metode eksak untuk VRP di kembangkan oleh Dantzig dan Ramser [6]. Dantzig dan Ramser [6] memodifikasi algoritma yang yang digunakan untuk TSP oleh Dantzig et al. [5]. Eilon et al. [8] mengembangkan pendekatan *Dynamic Programming* untuk VRP. Laporte dan Norbert [11] dan Laporte et al. [12] menggunakan metode *Branch and Bound*. Untuk metode heuristic/metaheuristik, algoritma *saving* Clarke dan Wright [4] menjadi basis dari banyak algoritma untuk menyelesaikan VRP Christofides dan Eilon [3] mengajukan metode *improvement* yang menggunakan 2-opt dan 3-opt yang awalnya dikembangkan oleh Lin [14] untuk TSP. Salhi dan Rand [19] mengajukan metoda heuristik yang menggunakan beberapa prosedur *refinement*. Osman [16] memakai *Simulated Annealing* dan *Tabu Search* metaheuristik dengan metoda $\lambda$ - *interchange*. Taillard [20] mengembangkan metoda yang mempartisi masalah berukuran besar sebelum mengaplikasikan *Tabu Search*. Xu dan Kelly [23] menggunakan *network flow-based Tabu Search*. Algoritma Neural Network digunakan Torki et al. [22] untuk VRP. Varian dari algoritma *threshold accepting* yaitu algoritma *Backtracking Adaptive Threshold Accepting* (BATA) dikembangkan oleh Tarantilis et al. [21]. Prins [18] menggunakan *Genetic Algorithm* (GA) untuk menyelesaikan VRP. Pisinger dan Ropke [17] mengaplikasikan *general heuristic* yang dapat menyelesaikan lima varian VRP. Algoritma *Honey Bee Mating* diajukan oleh Marinakis et al. [15]. Metoda *Improved Ant Colony Optimization* (IACO) dikembangkan oleh Yu et al. [24].

Pada makalah ini, bagian 2 menjelaskan algoritma usulan, bagian 3 menampilkan hasil perhitungan dan yang terakhir adalah kesimpulan dan saran.

## 2. Metode Penelitian

VND metaheuristik dikembangkan oleh Hansen dan Mladenovic [10] untuk menyelesaikan masalah optimisasi kombinatorik. VND merupakan varian yang bersifat deterministik dari Variable Neighborhood Search meheuristik. Ide dasar dari VND adalah perubahan *neighborhood* yang sistematis dalam metode pencarian lokal (*local search*). Penelitian ini merupakan peneliatian awal penggunaan VND untuk menyelesaikan VRP dan varian-variannya. AlgoritmaVND usulan dilengkapi dengan beberapa *local search* termasuk algoritma Dijkstra dan skema diversifikasi. Algoritma yang digunakan pada penelitian ini dapat dilihat pada Gambar 1.

*Initialization*. Select a set of neighbourhood structures $N_l$, for $k = 1,...,l_{\max}$ that will be used in the search; find an initial solution $x$; define the maximum number of diversifications, *NbDivMax*., set *NbDiv* = 0.

*Repeat* the following sequence until no improvement is obtained:

(1) Set $l \leftarrow 1$

(2) *Repeat* the following steps until $l = l_{\max}$ :

*(a) Exploration of neighbourhood.* Find the best neighbour $x'$ of

$$x(x' \in N_l(x));$$

*(b) Move or not.* If the solution $x'$ obtained is better than $x$,

set $(x \leftarrow x')$, dan $l \leftarrow 1$;

otherwise, set $l \leftarrow l+1$. If $l > l_{max}$ continue to (c)

*(c)* Construct the cost network using $x$ and apply Dijkstra's algorithm to get $x'$.

If the new solution $x'$ is better than $x$, set $(x \leftarrow x')$ and go to (1) else

go to (d)

*(d)* if $NbDiv > NbDivMax$ then stop, else set $NbDiv \leftarrow NbDiv + 1$, apply the

diversification procedure and

go to (1).

Gambar 1: Algoritma VND Usulan

**Solusi inisial**

Solusi inisial diperoleh dari tiga langkah; (a) Membuat satu *giant tour* menggunakan algoritma *sweep* [9], (b) perbaiki hasil langkah (a) menggunakan 2-opt [14], dan (c) bentuk *cost network* dan kemudian gunakan algoritma Dijkstra [7] untuk menemukan solusi optimal untuk urutan customer yang sekarang. Untuk menghindari menggunakan jarak terbesar antara dua *customer* berurutan di satu rute, titik awal dalam pembentukan *cost network* adalah titik titik yang mempunyai jarak terjauh antara dua *customer* (gap) di dalam *giant tour*. Jumlah gap (*NG*) yang di gunakan didefinisikan sebagai berikut:

$NG =$

$$Min\{\max(8, \tfrac{NR}{2}), |((i,i+1) : g_i > \min(\bar{g}, \tfrac{g^+}{2}))|\} \quad (1)$$

*NR* : jumlah rute yang didapat oleh algoritma Dijkstra's

$(i,i+1)$ : merupakan urutan konsumen

$g_i$ : gap ke $i$ (jarak antara konsumen $i$ dan $i+1$),

$\bar{g}$ : rata-rata gap

$g^+$ : gap terbesar.

Alasan penggunaan (1) didasarkan pada ide untuk menyambung nilai *NG* dengan jumlah rute dan jumlah gap yang berhubungan dengan rata-rata dan gap terbesar. Untuk setiap *NG* gap yang terpilih misalkan $(i_l, i_l+1)$, dua *cost network* dibangkitkan mulai dari $i_l$ berlawanan arah putaran jarum jam dan mulai dari $i_l+1$ searah putaran jarum jam Algoritma Dijkstra's kemudian dipakai pada setiap $2 \times NG$ *cost network*.

**Pembentukan *cost network***

Untuk mengaplikasikan algoritma Dijkstra, pertama dibuat *cost network* dengan memperhatikan data *customer*, pembatas kapasitas dan pembatas jarak kendaraan. Algoritma Dijkstra digunakan untuk

mendapatkan biaya terkecil dari jaringan yang berawal dari depot dan berakhir di simpul terakhir.

**Struktur *Neighborhood* (*Local Search*)**

Pada algoritma VND usulan enam *neighborhood* atau *local search* digunakan. Urutan penggunaan dari neighborhood-neighborhood tersebut adalah sebagai berikut: 1-*insertion intra-route* ($l_1$), 2-*insertion intra-route* ($l_2$), *swap intra-route* ($l_3$), 2-opt *inter-route* ($l_4$), 1- *insertion intra-route* ($l_5$) dan 2-opt *intra-route* ($l_6$). Proses dimulai dengan menggunakan solusi inisial sebagai input bagi $l_1$ , kemudian dicari solusi terbaik dari $l_1$, jika tidak ditemukan lagi solusi yang lebih baik di $l_1$ maka solusi tersebut digunakan sebagai input bagi $l_2$. Di $l_2$ dicari solusi yang terbaik, jika didapatkan solusi yang lebih baik daripada input dari $l_1$ , maka pencarian kembali ke $l_1$ dengan solusi yang lebih baik tersebut menjadi input untuk $l_1$ . Jika pada $l_2$ tidak ditemukan solusi yang lebih baik dari input $l_1$ maka pencarian diteruskan ke $l_3$ dengan input solusi terbaik dari $l_1$ . Proses yang sama diulang sampai $l_6$. Jika pada $l_6$ tidak ditemukan solusi yang lebih baik daripada input dari $l_5$ proses dilanjutkan dengan menggunakan algoritma Dijkstra. Jika dari algoritma Dijkstra diperoleh solusi yang lebih baik, pencarian kembali ke $l_1$ , jika tidak ditemukan solusi yang lebih baik proses diteruskan ke proses diversifikasi. Proses diversikasi digunakan sampai jumlah maksimum diversikasi maksimum dicapai. Seluruh proses pencarian berhenti jika jumlah maksimum diversifikasi telah dicapai.

Prosedur *1-insertion (intra-route dan inter-route)*

Dua jenis prosedur 1-*insertion* diterapkan. Yang pertama adalah 1-*insertion intra-route* dan yang kedua adalah 1-*insertion inter-route*. Pada 1-*insertion intra-route* satu *customer* dipindahkan dari tempatnya di suatu rute dan di coba disisipkan ditempat dalam suatu rute untuk memperoleh rute yang lebih baik. Sedangkan pada 1-*insertion inter-route*, setiap *customer* dari suatu rute dipindahkan ke rute yang lain. Jika perpindahan ini tidak melanggar pembatas yang ada maka *customer* yang terpilih akan dipindahkan.

*2-insertion (intra-route)*

2-*insertion intra-route* memungkinkan kita untuk memindahkan dua *customer* yang berurutan dan menyisipkan mereka ke suatu tempat dalam rute yang sama untuk mendapatkan rute yang lebih baik.

*Prosedur 2-opt (inter-route dan intra-route)*

2-opt *intra-route*, mempunyai nama lain 2-opt (lihat [14]), merupakan prosedur perbaikan yang efektif. Prosedur ini bekerja dengan menghilang dua *arc* yang tidak berdekatan pada satu rute dan menambahkan dua *arch* yang baru. Proses pertukaran dilakukan sampai perbaikan dari *total cost* tidak ditemukan lagi. 2-opt *inter-route* sama dengan 2-opt *intra-route* kecuali pada 2-opt *intra-route* dua rute diperhatikan dua rute dalam pertukaran.

*Prosedur swap (intra-route)*

*Swap intra-route* bertujuan untuk mengurangi *total cost* atau mendapatkan rute yang lebih baik dari suatu rute dengan cara menukar tempat dua *customer* dalam suatu rute tersebut.

**Penggunaan Algorithm Dijkstra sebagai Prosedur Perbaikan**

Disamping digunakan untuk membangkitkan solusi inisial, algoritma Dijkstra juga digunakan sebagai prosedur perbaikan. *Cost network* dibentuk dari solusi terbaik *incumbent*. Tujuannya adalah untuk melihat apakah solusi yang didapat sudah merupakan solusi optimum dari *cost network* yang dibuat.

**Prosedur Diversifikasi**

Prosedur ini digunakan jika setelah semua *local search* dilakukan tidak ditemukan lagi solusi yang lebih baik. Prosedur ini bertujuan untuk mengeksplorasi daerah ruang pencarian (*search space*) yang lain yang mungkin belum dikunjungi. Solusi terbaik *incumbent* digunakan sebagai untuk mendapatkan solusi inisial yang baru. Pada penelitian ini jumlah diversifikasi yang dilakukan (*ND*) ditentukan sebagai berikut:

$$ND = Min\,(100, 2N) \qquad (2)$$

N:jumlah*customer*.

Langkah-langkah prosedur diversifikasi adalah sebagai berikut:

1. Hubungkan semua simpul. Simpul terakhir dari rute awal disambungkan dengan simpul pertama dari simpul pertama rute berikutnya.

2. Hitung semua jarak antara dua simpul berurutan.

3. Pilih jarak terbesar antara dua simpul berurutan yang bukan merupakan ujung simpul dari

rute yang berbeda, sebut $(e_1, e_2)$ sebagai *starting point*.

4. Buat *cost network* mulai dari $e_2$ searah putaran jam dan gunakan algoritma Dijkstra.

5. Seperti langkah 4, tapi mulai dari $e_1$ berlawanan arah putaran jarum jam.

### 3. Hasil Perhitungan

Untuk proses pengujian, algoritma usulan diprogram ke dalam bahasa C++ dan digunakan untuk menyelesaikan data set [2]. Program dieksekusi menggunakan komputer dengan prosesor Intel M 1.7 GHz PC dan 1GB RAM. Solusi yang diperoleh kemudian dibandingkan dengan solusi-solusi yang telah dipublikasikan.

Tabel 1: Kualitas Solusi VND

| No | Size | Best Solution | Osman [16] | Taillard [20] | Xu & Kelly [23] | Barbazoglu & Ozgur [1] | Tarantilis et al. [21] | Prins [18] | Yu et al. [24] | VND |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 50 | 524.61 | 534.61 | 524.61 | 524.61 | 524.61 | 524.61 | 524.61 | 524.61 | 524.61 |
| 2 | 75 | 835.26 | 844.00 | 835.26 | 835.26 | 836.71 | 838.13 | 835.26 | 835.26 | 854.04 |
| 3 | 100 | 826.14 | 838.00 | 826.14 | 826.14 | 826.72 | 830.69 | 826.14 | 830.00 | 843.46 |
| 4 | 150 | 1028.42 | 1044.35 | 1028.42 | 1029.14 | 1043.89 | 1036.38 | 1030.46 | 1028.42 | 1063.42 |
| 5 | 199 | 1291.45 | 1334.55 | 1291.45 | 1298.58 | 1306.16 | 1317.81 | 1296.39 | 1305.50 | 1368.65 |
| 11 | 120 | 1042.11 | 1043.00 | 1042.11 | 1042.11 | 1051.18 | 1042.13 | 1042.11 | 1042.11 | 1169.10 |
| 12 | 100 | 819.56 | 819.59 | 819.56 | 819.56 | 819.56 | 819.5... | ...9.56 | 819.56 | 842.75 |
| # Best Solution | | | 1 | 7 | 5 | 2 | 3 | 5 | 5 | 1 |
| Average Deviation (%) | | | 1.065 | 0.000 | 0.099 | 0.571 | 0.530 | 0.083 | 0.222 | 4.106 |

Dari Tabel 1 terlihat algoritma usulan menghasilkan 1 solusi yang sama dengan solusi terbaik. Deviasi rata-rata yang dihasilkan tidak sebaik deviasi rata-rata yang telah dipublikasikan. Pada Tabel 2 dapat dilihat bahwa algoritma usulan menggunakan waktu komputasi (*CPU Time*) yang *acceptable*.

Tabel 2: Perbandingan CPU *Time* (dalam detik)

| No | Size | Osman | Taillard | Xu dan Kelly | Barbazoglu dan Ozgur | Tarantilis et al. | Prins | Yu et al. | VND |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 50 | 114 | 49 | 30 | 455 | 122 | 1 | 2 | **11** |
| 2 | 75 | 50 | 53 | 49 | 2401 | 132 | 46 | 11 | **48** |
| 3 | 100 | 1543 | 580 | 72 | 2040 | 189 | 28 | 30 | **149** |
| 4 | 150 | 3560 | 3800 | 150 | 4604 | 385 | 330 | 211 | **409** |
| 5 | 199 | 3246 | 3000 | 273 | 7595 | 1059 | 1147 | 677 | **962** |
| 11 | 120 | 1445 | 4600 | 57 | 4214 | 247 | 18 | 61 | **226** |
| 12 | 100 | 892 | 340 | **91** | 2277 | **65** | 3 | **31** | **114** |

### 4. Simpulan dan Saran

Adaptasi awal dari algoritma dasar VND telah dikembangkan untuk menyelesaikan VRP. Algoritma VND dasar dilengkapi dengan beberapa *local search*, algoritma Dijkstra dan prosedur diversifikasi. Dari uji coba menggunakan data set yang ada pada literatur, algoritma usulan menghasilkan solusi yang kualitasnya di bawah beberapa solusi yang telah dipublikasikan. Artinya metoda usulan memerlukan perbaikan untuk dapat menghasilkan solusi yang lebih baik.

Untuk penelitian lebih lanjut, algoritma VND usulan akan diperbaiki dengan menggunakan *local search- local search* yang lain seperti Or-opt, 3-opt, GENI dan 2-1 insertion. Algoritma ini juaga akan diaplikasikan untuk menyelesaikan masalah-masalah VRP lainnya seperti *heterogeneous fleet* VRP, *multi-depot* VRP dan *heterogeneous fleet multi-depot* VRP.

### Daftar Pustaka


[1] Barbazoglu, G. dan Ozgur, D. (1999). A Tabu Search for the Vehicle Routing Problem. *Computers & Operations Research* **26**, 255-270.

[2] Christofides, N. (1976). The Vehicle Routing Problem. *Recherche Operationnelle* **10**, 55-70.

[3] Christofides, N. dan Eilon, S. (1969). An Algorithm for the Vehicle Dispatching Problem. *Operational Research Quarterly* **20**, 309-318.

[4] Clarke, G. dan Wright, J.W. (1964). Scheduling of Vehicle from Central Depot to a Number of Delivery Points. *Operations Research* **12**, 568-581.

[5] Dantzig, G., Fulkerson, D. dan Johnson, S. (1954). Solution of Large

Scale Travelling Salesman Problem. *Operations Research* **2**, 393-410.

[6] Dantzig, G. dan Ramser, J. (1959). The Truck Dispatching Problem. *Management Science* **6**,80-91.

[7] Dijkstra, E.W. (1959). A Note on Two Problems in Connection with Graphs. *Numerische Mathematik* **1**, 269-271.

[8] Eilon, S., Watson-Gandy, C. dan Christofides, N. (1971). *Distribution Management: Mathematical Modelling & Practical Analysis.* Griffin, London.

[9] Gillet, B.E. dan Miller, L.R. (1974). A Heuristic Algorithm for the Vehicle Dispatch Problem. *Operations Research* **22**, 340-344.

[10] Hansen, P. dan Mladenovic, N. (2003). A Tutorial on Variable Neighbourhood Search. *Le Cahiers du GERAD G-2003-46.*

[11] Laporte, G. dan Nobert, Y. (1987). Exact Algorithm for the Vehicle Routing Problem. *Annals of Discrete Mathematics* **31**, 147-184.

[12] Laporte, G., Mercure, H. dan Nobert, Y. (1992). A Branch-and-Bound Algorithm for a Class of Asymmetrical Vehicle Routeing Problems. *Journal of Operational Research Society* **43**, 469-481.

[13] Lenstra, J.K. dan Rinnooy Kan, A.H.G. (1975). Some Simple Applications of the Travelling Salesman Problem. *Operational Research Quarterly* **26**, 717-734.

[14] Lin, S. (1965). Computers Solutions of the Travelling Salesman Problem. *Bell System Technical Journal* **44**, 2245-2269.

[15] Marinakis, Y., Marinaki, M., dan Dounias, G. (2008).Honey Bees Mating Optimization Algorithmfor the Vehicle Routing Problem. *Studies in Computational Intelligences*, **129**, 139-148.

[16] Osman, I.H. (1993). Metastrategy Simulated Annealing dan Tabu Search Algorithms for Vehicle Routing Problem. *Annals of Operations Research* **41**, 421-451.

[17] Pisinger, D. dan Ropke, S. (2007). A General Heuristic for Vehicle Routing Problem. *Computers & Operations Research* **34**, 2403-2435.

[18] Prins, C. (2004). A Simple dan Effective Evolutionary Algorithm for the VehicleRouting Problem. *Computers & Operations Research* **31**, 1985-2002. Boston.

[19] Salhi, S. dan Rand, G.K. (1987). Improvements to Vehicle Routing Heuristics. *Journal of the Operational Research Society* **38**, 293-295.

[20] Taillard, E.D. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks* **23**, 661-676.

[21] Tarantilis, C.D., Kiranoudis, C.T., dan Vassiliadis, V.S., (2002). A Backtracking Adaptive Threshold Accepting Metaheuristic Method for the Vehicle Routing Problem. *System Analysis Modelling Simulation* **42**, 631-664.

[22] Torki, A., Somhon, S. dan Enkawa, T. (1997). A Competitive Neural Network Algorithm for Solving Vehicle Routing Problems. *Computers & Industrial Engineering* **31**,473-476.

[23] Xu, J. dan Kelly, J.P. (1996). A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. *Transportation Science* **30**, 379-393.

[24] Yu, B., Yang, Z. dan Yao, B. (2009). An Improved Ant Colony Optimization for Vehicle Routing Problem. *European Journal of Operational Research* **196**, 171-176.