Computational Comparison of Alternative Strategies with Interior Point Methods for Network Piecewise Linear Programs

Fernando A. S. Marins – FEG – UNESP – Brazil – fmarins@feg.unesp.br

Clovis Perin – IMECC – UNICAMP – Brazil – clovis@ime.unicamp.br

Margarida Mello – IMECC – UNICAMP – Brazil – margarid@ime.unicamp.br

Resumo

A abordagem usual de solução de um problema de fluxo em rede linear por partes é a sua transformação em um problema equivalente de fluxo em rede linear. Nesta transformação, uma rede linear por partes com *n* nós e *m* arcos, cada um com ℓ_j intervalos para cada parte linear da função de custo associada ao arco j, tem uma rede linear associada com ***n*** nós e $\ell = \sum_{j=1}^{m} \ell_j$ arcos. Métodos de pontos interiores têm se apresentado muito bem na resolução de problemas de fluxo em redes lineares. Nós mostramos que é vantajoso construir um método de pontos interiores especializado para resolver problemas em redes lineares por partes diretamente, ao invés de aplicar uma versão genérica no problema linear equivalente. Dois algoritmos foram implementados e testados: um utilizando o passo preditor-corretor e o outro sem utilizar o passo corretor. Comparações entre diferentes estratégias alternativas (inicialização e critério de parada) são realizadas para efeito de testes computacionais

Computational Comparison of Alternative Strategies with Interior Point Methods for Network Piecewise Linear Programs

Fernando A. S. Marins – FEG – UNESP – Brazil – fmarins@feg.unesp.br

Clovis Perin – IMECC – UNICAMP – Brazil – clovis@ime.unicamp.br

Margarida Mello – IMECC – UNICAMP – Brazil – margarid@ime.unicamp.br

Abstract

The usual approach to solving a piecewise linear network flow problem is to transform it into an equivalent linear one. In this transformation, a piecewise network with *n* nodes and *m* arcs, each with ℓ_j intervals to each linear piece of the cost function associated to arc j, has an equivalent linear one with ***n*** nodes and $\ell = \sum_{j=1}^{m} \ell_j$ arcs. Interior point methods have been proved successful in the solution of linear network flow problems. We show that is advantageous to construct a customized interior point method to solve piecewise linear network problems directly, instead of applying its generic version to the equivalent linear problem. Two algorithms were implemented and tested: one using predictor-corrector and the other without the corrector step. Comparison between alternative strategies (initialization and stopping criteria) is done by means of several computational tests.

Keywords: Piecewise linear network, primal – dual interior point method, predictor – corrector, computational tests.

1. Introduction

An important area of Mathematical Programming is the Piecewise Linear Programming, which is related with the optimization of a separable convex piecewise linear objective function. In this paper, it is given a special attention to a sub-area known as Piecewise Linear Network Programming, which

has relevant application in the real world, like in: power systems, pipeline networks, transportation, communication networks, water management and stochastic network flow problems. In [2] the authors investigated the relative performance of specialized algorithms for piecewise linear network problems: (Strongly Feasible) Primal Simplex, Dual Simplex, Out-of-kilter and (Strongly Polynomial) Cost-Scaling algorithms.

We gave sequence to the research related in [6]. In that paper, the implementation of an adaptation of an interior point method for piecewise linear flow problems was described. That implementation is now modified, incorporating improvements proposed by [5], specifically in what concerns with the utilization of a predictor-corrector scheme. It must be said that the usual implementation of the predictor-corrector method, as described in [5], utilize the Cholesky's Decomposition, as for example in [1,4]. In our implementation, however, we solve the linear systems, associated to predictor-corrector method, by a conjugated gradient method, which takes advantage of the problem's incidence matrix, as proposed by [7] and shown in [6].

This paper is organized as follows. The algorithms are presented in next section, their implementations are the object of section 3, and the computational tests are related and commentated in section 4. The Tables and Figures are in the Appendix at the end of the paper.

2. Algorithms

The piecewise linear network flow problem is defined by

$$\text{Min}\{f(x) \text{ such that } Ax = b,\ x \geq 0\},$$

where A is a *n x m* - incidence matrix of a network with *n* nodes and *m* arcs, *b* is the *n-vector* of demand in the nodes, *x* is the *m*-vector of the flows in the arcs (to be determined), and $f(x) = \sum_{j=1}^{m} f_j(x_j)$ is a separable convex piecewise linear function. The convex piecewise linear functions f_j can be specified by the pair of ℓ-vectors *c* and *d* where $\ell = \sum_{j=1}^{m} \ell_j$ is the total number of intervals, and ℓ_j is the number of intervals of f_j.

This problem can be transformed into a linear network flow problem (see [3]) defined by:

$$\min\left\{c'\bar{x} \text{ such that } \bar{A} = b, \bar{x} + \bar{s} = d, \bar{x}, \bar{s} \geq 0\right\}$$

In this case, the *n* x ℓ - matrix $\bar{A}$ is obtained from the matrix A by repeating its columns as many times as are the intervals of the associated arcs. We are supposing that this same procedure is also made in vector *x* transforming it in the vector $\bar{x}$. The associated dual problem is given by

$$\max\left\{b'y - d'\bar{w} \;\text{ such that }\; \bar{A}'y - \bar{z} + \bar{w} = c, \bar{z}, \bar{w} \geq 0\right\}$$

and the complementary slack conditions are expressed by

$$\bar{X}\,\bar{Z}\,\mathbf{1} = 0 \quad \text{and} \quad \bar{S}\,\bar{W}\,\mathbf{1} = 0,$$

where $\bar{X}, \bar{S}, \bar{Z}, \bar{W}$ are diagonal matrixes with the elements of $\bar{x}, \bar{s}, \bar{z}, \bar{w}$, respectively, and **1** is the vector of 1's**.**

Two versions of interior point methods were studied: the usual primal-dual (pure predictor) and the primal-dual with predictor-corrector.

As described in [6], the usual primal-dual interior point method is initialized from $\bar{x}, \bar{s}, y, \bar{z}, \bar{w}$**,** such that $\bar{x}, \bar{s}, \bar{z}, \bar{w} > 0$ and at each iteration a search direction $\Delta\bar{x}, \Delta\bar{s}, \Delta\bar{y}, \Delta\bar{z}, \Delta\bar{w}$ is obtained, as a function of a centering parameter **μ**, and according to the equations below that reflect the primal feasibility, the dual feasibility**,** and the complementary slackness conditions when $\mu = 0$:

$$\begin{aligned}
\bar{A}\Delta\bar{x} &= b - \bar{A}\bar{x} \\
\Delta\bar{x} + \Delta\bar{s} &= d - \bar{x} - \bar{s} \\
\bar{A}'\Delta y + \Delta\bar{z} - \Delta\bar{w} &= c - \bar{A}'y - \bar{z} + \bar{w} \\
\bar{X}\Delta\bar{z} + \bar{Z}\Delta\bar{x} &= \mu\mathbf{1} - \bar{X}\bar{z} \\
\bar{S}\Delta\bar{w} + \bar{W}\Delta\bar{s} &= \mu\mathbf{1} - \bar{S}\bar{w}
\end{aligned} \qquad (1)$$

The usual primal-dual interior point method can be described as it follows:

let $\bar{x}, \bar{s}, y, \bar{z}, \bar{w}$ **, such that** $\bar{x}, \bar{s}, \bar{z}, \bar{w} > 0$

while (stopping criteria is not satisfied) **do**

update μ

obtain $\Delta\bar{x}, \Delta\bar{s}, \Delta\bar{y}, \Delta\bar{z}, \Delta\bar{w}$

find α **such that** $(\bar{x}, \bar{s}, \bar{z}, \bar{w}) + \alpha(\Delta\bar{x}, \Delta\bar{s}, \Delta\bar{z}, \Delta\bar{w}) > 0$

update $(\bar{x}, \bar{s}, y, \bar{z}, \bar{w}) \leftarrow (\bar{x}, \bar{s}, y, \bar{z}, \bar{w}) + \alpha(\Delta\bar{x}, \Delta\bar{s}, \Delta\Delta y, \bar{z}, \Delta\bar{w})$

On the other hand, in the primal-dual interior point method with predictor-corrector both systems (1) and (2) have the same coefficient matrix on the left hand side:

$$\begin{aligned}
\bar{A}\Delta\bar{x} &= b - \bar{A}\bar{x} \\
\Delta\bar{x} + \Delta\bar{s} &= d - \bar{x} - \bar{s} \\
\bar{A}'\Delta y + \Delta\bar{z} - \Delta\bar{w} &= c - \bar{A}'y - \bar{z} + \bar{w} \\
\bar{X}\Delta\bar{z} + \bar{Z}\Delta\bar{x} &= \mu\mathbf{1} - \bar{X}\bar{z} - \delta\bar{X}\delta\bar{z} \\
\bar{S}\Delta\bar{w} + \bar{W}\Delta\bar{s} &= \mu\mathbf{1} - \bar{S}\bar{w} - \delta\bar{S}\delta\bar{w}
\end{aligned} \qquad (2)$$

where $\delta\bar{x}, \delta\bar{z}, \delta\bar{s}, \delta\bar{w}$ correspond to the solution of the system (1).

The $(\Delta\bar{x}, \Delta\bar{s}, \Delta\bar{z}, \Delta\bar{w})$ vector is the solution of (1) for the pure predictor algorithm and it is the solution of (2) for the algorithm with predictor-corrector. The α parameter is obtain by a usual ratio test for the pure predictor algorithm and by means a special scheme, as described in [1, 5] for the predictor-corrector algorithm.

We utilized the strategy of solving each one of the above predictor and corrector systems of equations by means a Conjugated Gradient Method applied to *n* x *n* – symmetric positive definite system (SPD). The matrix associated to that system is implicitly used when we traverse all the arcs of problem's network. These symmetric positive definite systems are solved with the conjugated gradient method with diagonal preconditioning for the six first iterations of the primal-dual method and with the preconditioning of the spanning tree in the subsequent iterations.

The stopping criteria adopted are: (1) maximum number of iterations exceeded; (2) quasi--complementary solutions found ($\bar{x}'\bar{z} + \bar{s}'\bar{w} \approx 0$), (3) inferior limit value for the primal objective function exceeded; and (4) superior limit value for the dual objective function exceeded.

3- Implementation

The C^{++} language was used in the implementation of the algorithms. Four pairs of programs associated with different versions of the interior point method were implemented and tested; four of them for linear network and another four for piecewise linear network. One pair (FL9, FP9) is the usual pure predictor which solves only one symmetric positive definite system in each iteration of the primal-dual method, and three pairs (FL0- FP0, FL1-FP1, FL2-FP2) are of the predictor-corrector kind, those solve two symmetric positive definite systems in each iteration of the primal-dual method.

The corrector system is built in a way that its solution is search direction of the iterate; in that way, we apply a conjugated gradient method using the predictor system's solution as a starting solution for the corrector system.

The starting solution of the predictor system is the null solution, and the starting solution of the corrector system is the final solution of the first system for two pairs of implemented programs (FL0-FP0 and FL1-FP1), and is the null solution for one pair (FL2, FP2) of implemented programs.

The stopping criterion for the conjugated gradient method always uses a tolerance ε_p or ε_c for the relative error of the current solution in the predictor or corrector systems, respectively. In short, the implemented programs are:

- ✓ FL9 – pure predictor for the linear network with a $\varepsilon_p = 10^{-8}$;

- ✓ FL0 – predictor-corrector for the linear network with $\varepsilon_p = \varepsilon_c = 10^{-8}$, uses the predictor system's final solution as the corrector system's starting equation;
- ✓ FL1 – predictor-corrector for the linear network with $\varepsilon_p = 10^{-6}$ and $\varepsilon_c = 10^{-8}$; uses the predictor system's final solution as the corrector system's starting solution;
- ✓ FL2 – predictor-corrector for the linear network with $\varepsilon_p = \varepsilon_c = 10^{-8}$; uses the null solution to start the corrector system;
- ✓ FP9 – pure predictor for the piecewise linear networks with $\varepsilon_p = 10^{-8}$;
- ✓ FP0 – predictor-corrector for the piecewise linear networks with $\varepsilon_p = \varepsilon_c = 10^{-8}$, uses the predictor system's final solution as the corrector system's starting equation;
- ✓ FP1 – predictor-corrector for the piecewise linear networks with $\varepsilon_p = 10^{-6}$ and $\varepsilon_c = 10^{-8}$, uses the predictor system's final solution as the corrector system's starting equation;
- ✓ FP2 – predictor-corrector for the piecewise linear networks with $\varepsilon_p = \varepsilon_c = 10^{-8}$; uses the null solution to start the corrector system.

The memory space required by these implemented programs to store the network data, the current solutions, the conjugated gradient's solutions and other auxiliary variables can be summarized in:

- ✓ FL9 - 13 *n*-vectors and 16 ℓ-vectors;
- ✓ FL0, FL1, FL2 - 15 *n*-vectors and 19 ℓ-vectors;
- ✓ FP9 - 13 *n*-vectors, 4 *m*-vectors and 13 ℓ-vectors;
- ✓ FP0, FP1, FP2 - 15 *n*-vectors, 4 *m*-vectors and 16 ℓ-vectors.

The tolerances used in the implementation were: 10^{-8} (complementary test), -10^{-8} (inferior limit of the primal objective function) e 10^{-8} (superior limit of the dual objective function).

4- Computational Tests

The algorithms were tested using a Pentium III 500 MHz, with 392 Mb of RAM memory**,** using connected network problems randomly generated with up to 100,000 nodes, 1,000,000 arcs and 1,000,000 million intervals.

The generated networks correspond to modified transportation problems. To the bipartite digraph's structure with equal number of supply and demand nodes, a directed cycle containing all supply nodes and other cycle containing all demand nodes were added, with the purpose to increase the probability of obtaining feasible generated problems. All the nodes of the generated network have the

same degree. The supply/demand, the cost coefficients and the arc's capacities were generated from a uniform distribution.

The CPU time (in seconds), the primal-dual method's number of iterations and the conjugated gradient's total number of iterations executed by each implemented algorithm for the generated networks (five networks generated with the same parameters and associated with five different seeds) are in Tables shown in the Appendix.

4.1. Initial Instances

First of all, relatively small problems were tested varying the parameters given to the generator, in order to seek to define directions for this study. The parameters involved are the interval for the supply/demand in the nodes, the interval for the cost coefficients, the interval for the arc's capacities, and the total number of nodes, of arcs and of intervals . The most promising data is presented in the Table 2 with supply/demand, cost and capacities randomly generated within a uniform distribution in the interval (0,100).

Table 3, shows the CPU times obtained with the nine problems groups, which were solved with the eight implemented programs versions. These data are also presented in Figure 1 (FLO, FL1, FL2, FP0, FP1 e FP2) and in Figure 2 (FL9, FL0, FP9 and FP0).

It can be established the superior performance of the specialized versions for piecewise linear networks. In each one of the two predictor-corrector classes, the type 1 versions (FP1, FL1) are slightly superior to the rest (see Figure 1). In Figure 2, it can observed that the predictor-corrector FP0 version shows to be better than almost all the other examples, with the exception of the G9 group (FL0, FP0 implementations) where it is beaten by the pure-predictor FP9 version.

4.2. Varying the number of intervals

Table 4 presents the CPU times obtained with the eight implemented algorithms to solve generated instances with 10,000 nodes, 35,000 arcs and the number of intervals varying from 70,000 up to 1,050,000. These data establish a superior performance of the specialized versions for piecewise linear networks. Besides that, FP9 performs better than almost all the other instances (see Figures 3 and 4).

4.3. Varying the number of arcs

Table 5 shows the CPU times obtained with the eight implemented programs in problems generated with 10,000 nodes and the varying the number of arcs from 20,000 up to 160,000. These data confirm a superior performance of the piecewise linear network specialized versions. Besides that, FP9 showed the best performance among almost all the other programs (see Figures 5 and 6).

4.4. Varying the number of nodes

Table 6 shows the CPU times obtained with the eight implemented algorithms in generated instances with 50,000 arcs, 25,000 intervals and varying the number of nodes between 5,000 and 20,000. These data establish a superior performance of piecewise linear network specialized version. In these tests, the predictor-corrector versions, specially FP1 showed a superior behavior among all the other versions (see Figures 7 and 8).

4.4. Varying the basic network

Table 7 shows the CPU times obtained with the eight implemented algorithms in generated instances with seven arcs per node and five intervals per arc whose number of nodes varying between 5,000 and 40,000. These data establish a superior performance of piecewise linear network specialized versions. In these tests, the predictor-corrector versions, specially FP1 showed a superior behavior among all the other versions (see Figures 9 and 10).

5. Final Considerations

The performed experiments also considered the total iteration's number associated with primal-dual method (see Table 8) and the total iteration's number associated with the conjugate gradient method with diagonal preconditioning and with the maximum generating tree preconditioning (see Table 9).

In a general way, the iteration's number of the usual primal-dual method version (or pure-predictor) was at least 50%, at most three times for some instances, superior to the iteration's number of the primal-dual method with predictor-corrector.

Besides this, it was verified that for bigger networks is better to use the diagonal preconditioning beyond the six initial iterations, as it was performed in general during the computational experiments related here.

Among the predictor-corrector versions, there was not a significant difference in the iteration's number of the primal-dual methods. Other types of variation in the structure and in the network coefficients were tested; however, the presented results were considered as the most interesting ones.

Summarizing, the pure-predictor version seems to have a better behavior only when the network has a large number of arcs or intervals in comparison to the number of nodes.

References

[1] J. Czyzyk, S. Mehrotra, M. Wagner and S. J. Wright, PCx user guide, Technical Report OTC 96/01. Optimization Technology Center, 1997.

[2] K. Darby-Dowman, F. A. S. Marins, E. L. F. Senne, C. Perin and A. Machado, Algorithms for network piecewise linear programs: a comparative study, *European Journal of Operational Research* 97 (1997), 183-199.

[3] J. K. Ho, Relationships among linear formulations of separable convex piecewise linear programs, *Mathematical Programming Study* 24 (1985), 126-140.

[4] M. Kojima, N. Megiddo and S. Mizuno, A Primal-Dual Infeasible-Interior Point Algorithm for Linear Programming, *Mathematical Programming* 61 (1993), 263-280.

[5] S. Mehrotra, On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization* 2 (1989), 575-601.

[6] C. Perin, M. P. Mello and F. A. S. Marins, An Implementation of Interior Point Method for Piecewise Linear Networks, *Tendências em Matemática Aplicada e Computacional* 1 (2000), No. 2, 431-442 (paper written in Portuguese and published by Brazilian Society of Computational and Applied Mathematics).

[7] M. Resende and G. Veiga, An efficient implementation of a network interior point method, Technical Report, AT&T Bell Laboratories, Murray Hill, NJ, USA, 1992.

Acknowledgements

To CNPq and FUNDUNESP for financial support.

APPENDIX

Program	*Symbol*	*Line*
FL9	•	dotted
FL0	○	dotted
FL1	▷	dotted
FL2	◇	dotted
FP9	•	full
FP0	○	full
FP1	▷	full
FP2	◇	full

Table 1. *Graphic Conventions*

Group	*Nodes*	*Arcs*	*Intervals*
G1	10 ,000	20,000	40,000
G2	10 ,000	20,000	100,000
G3	10 ,000	20,000	160,000
G4	10 ,000	35,000	70,000
G4	10 ,000	35,000	105,000
G6	10 ,000	35,000	280,000
G7	10 ,000	50,000	100,000
G8	10 ,000	50,000	250,000
G9	10 ,000	50,000	400,000

Table 2. *Groups of Initial Instances*

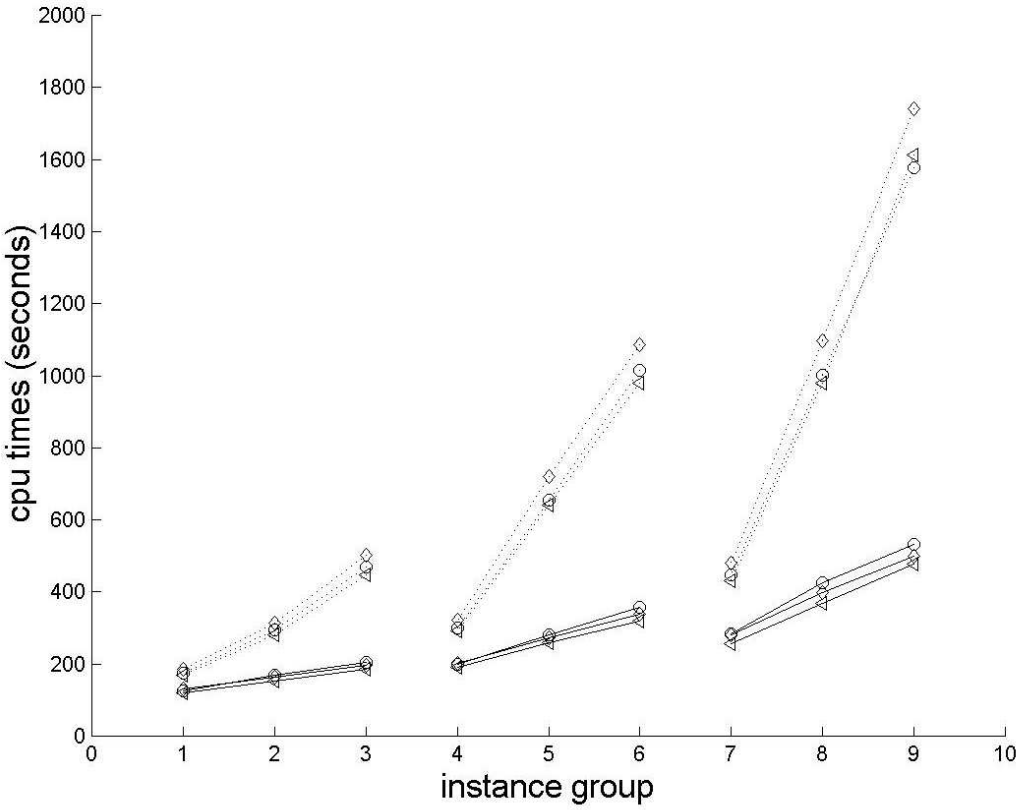

Figure 1. *CPU Times for the Predictor-Corrector Versions*

Groups	***G1***	***G2***	***G3***	***G4***	***G5***	***G6***	***G7***	***G8***	***G9***
Nodes(10^3)	***10***	***10***	***10***	***10***	***10***	***10***	***10***	***10***	***10***
Arcs (10^3)	***20***	***20***	***20***	***35***	***35***	***35***	***50***	***50***	***50***
Inter.(10^3)	***40***	***100***	***160***	***70***	***105***	***280***	***100***	***250***	***400***
FL9	988	859	772	913	674	636	857	649	775
FL0	175	295	468	301	655	1014	447	1002	1576
FL1	169	280	447	291	640	980	432	980	1613
FL2	186	313	502	322	721	1085	479	1097	1740
FP9	846	637	513	744	456	369	670	409	398
FP0	125	168	205	200	281	358	284	425	533
FP1	120	154	186	192	258	318	256	367	478
FP2	130	165	197	202	274	338	281	398	498

Table 3. *CPU seconds of the Initial Instances*

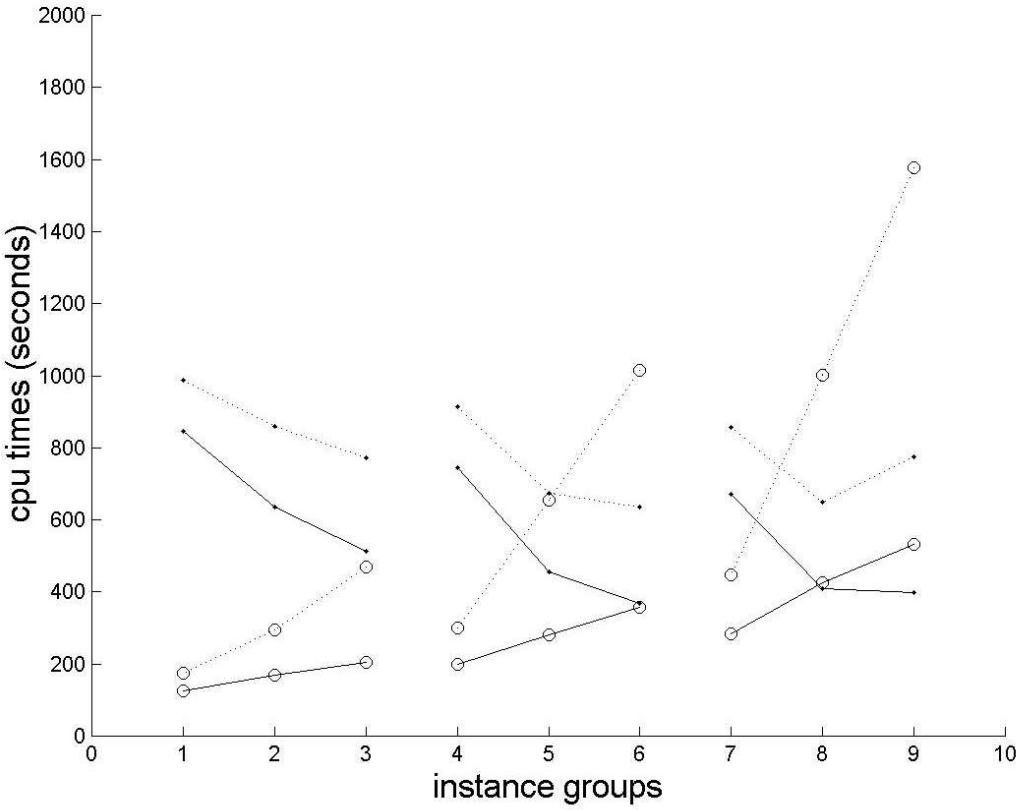

Figure 2. *CPU Seconds for FL9, FL0, FP9, FP0*

intervals	***70,000***	***175,000***	***280,000***	***525,000***	***1,050,000***
FL9	913	674	636	877	1806
FL0	301	655	1014	1784	4125
FL1	291	640	980	1918	4390
FL2	322	721	1085	1995	4640
FP9	744	456	369	**382**	**529**
FP0	200	281	358	461	702
FP1	**192**	**258**	**318**	457	687
FP2	202	274	338	446	705

Table 4. *CPU Seconds for different number of Intervals*

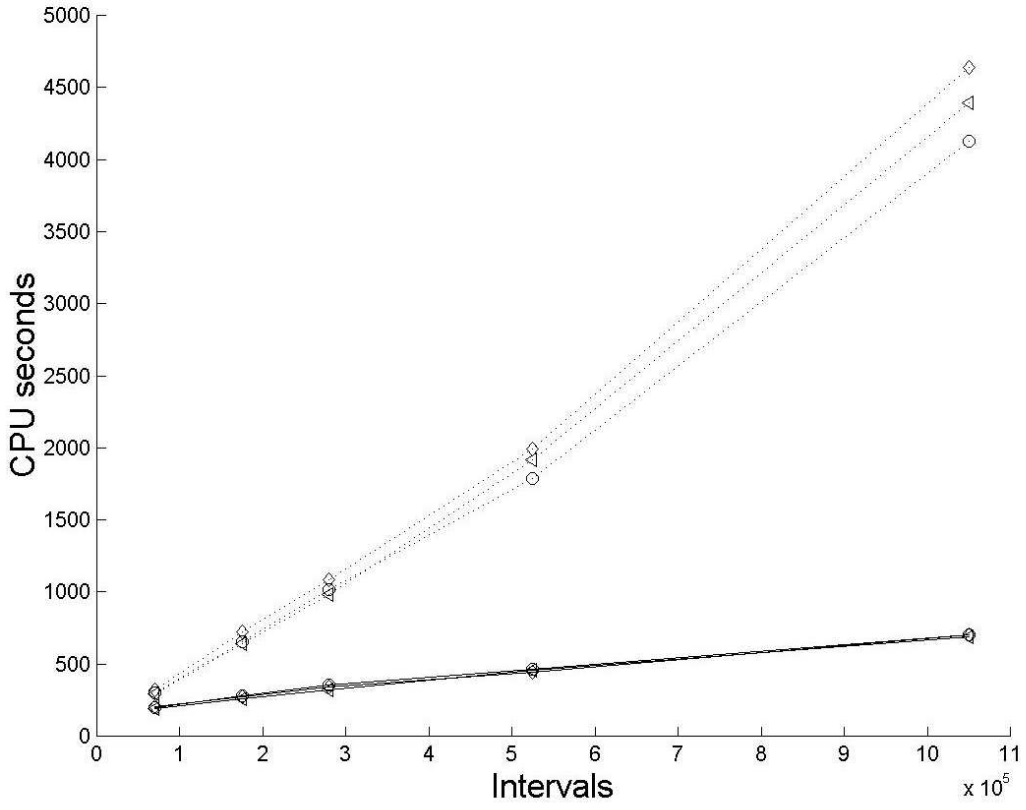

Figure 3. *CPU Seconds for different number of Intervals*

arcs	***20,000***	***35,000***	***50,000***	***85,000***	***160,000***
FL9	859	674	649	913	1970
FL0	305	697	1077	2179	5389
FL1	280	640	980	1997	4978
FL2	313	721	1097	2259	5746
FP9	637	456	409	485	878
FP0	161	271	398	736	1587
FP1	154	258	367	675	1471
FP2	165	274	398	746	1647

Table 5. *CPU Seconds for different number of Arcs*

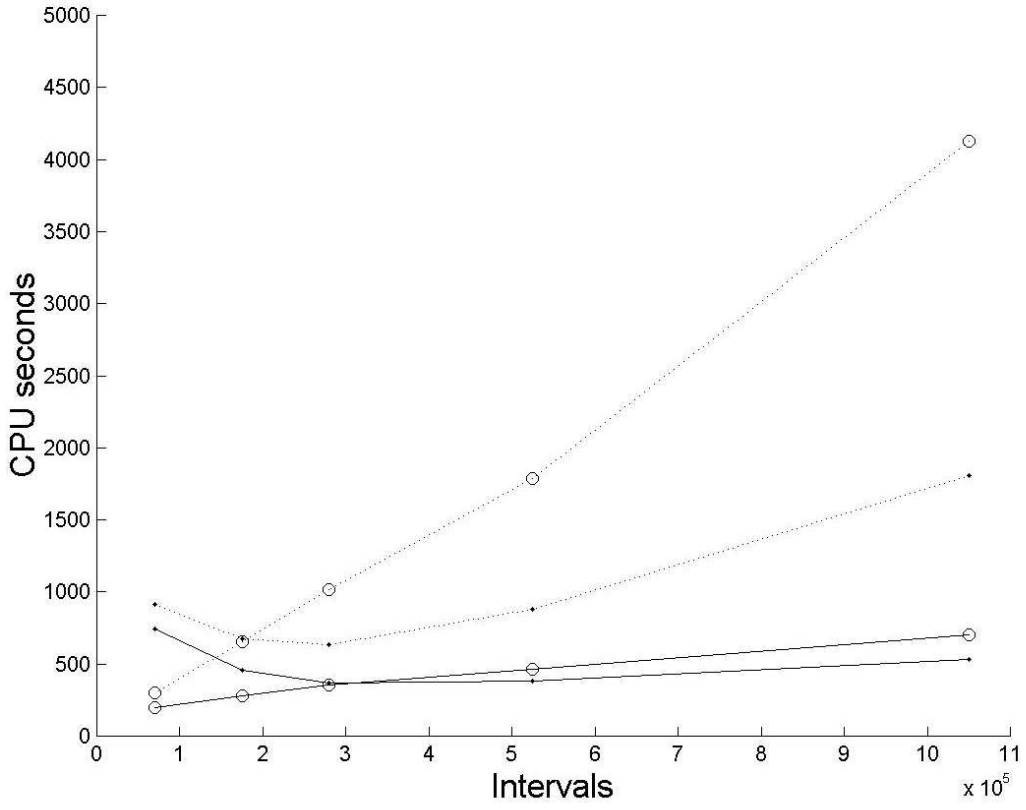

Figure 4. *CPU Seconds for different number of Intervals*

nodes	***5,000***	***10,000***	***125,000***	***20,000***
FL9	334	649	970	3120
FL0	798	1077	1171	1450
FL1	737	980	1090	1339
FL2	832	1097	1200	1496
FP9	**165**	409	650	2371
FP0	248	398	472	735
FP1	236	**367**	**442**	**695**
FP2	249	398	480	739

Table 6. *CPU Seconds for different number of Nodes*

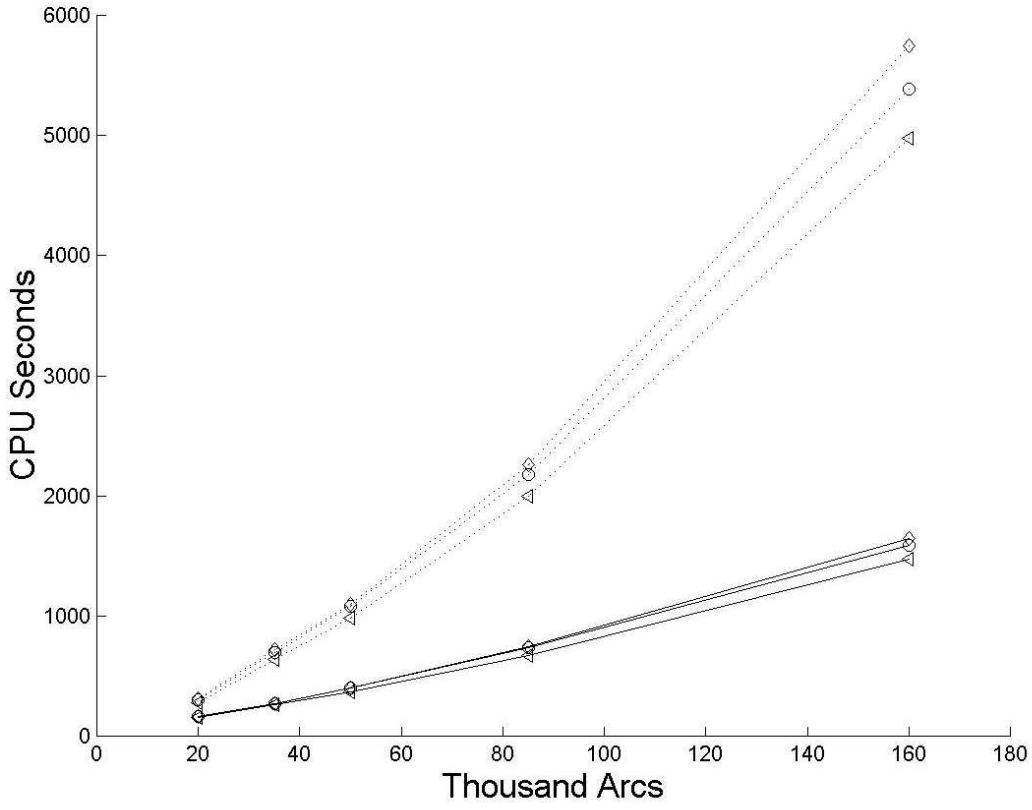

Figure 5. *CPU Seconds for different number of Arcs*

nodes	***5,000***	***10,000***	***20,000***	***30,000***	***40,000***
arcs	***35,000***	***70,000***	***140,000***	***210,000***	***280,000***
Intervals	***175,000***	***350,000***	***700,000***	***1,050,000***	***1,400,000***
FL9	183	674	2470	5815	11006
FL0	211	697	2512	5340	10021
FL1	193	640	2301	5015	9382
FL2	216	721	2553	5602	10562
FP9	115	456	1747	4227	8337
FP0	78	271	996	2219	4082
FP1	73	258	959	2183	3971
FP2	79	274	1010	2242	4132

Table 7. *CPU Seconds for Networks of different sizes*

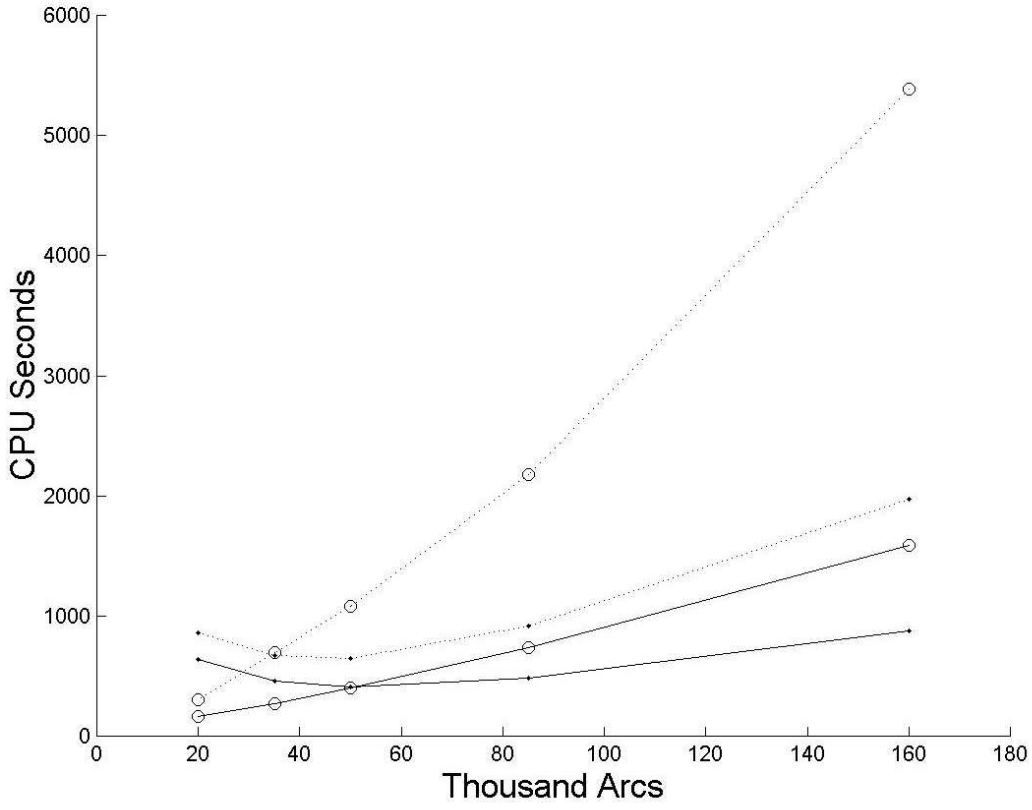

Figure 6. *CPU Seconds for different number of Arcs*

Groups	***G1***	***G2***	***G3***	***G4***	***G5***	***G6***	***G7***	***G8***	***G9***
FL9	188	142	110	149	88	66	123	69	61
FL0	25	28	30	29	34	37	32	38	42
FL1	25	27	30	29	35	38	33	38	43
FL2	25	28	30	30	35	37	32	39	43
FP9	189	142	110	149	88	66	123	69	61
FP0	25	27	30	29	33	37	32	38	42
FP1	24	27	30	29	33	37	32	38	43
FP2	25	28	30	29	34	37	32	38	43

Table 8. *Iterations of the Primal Dual*

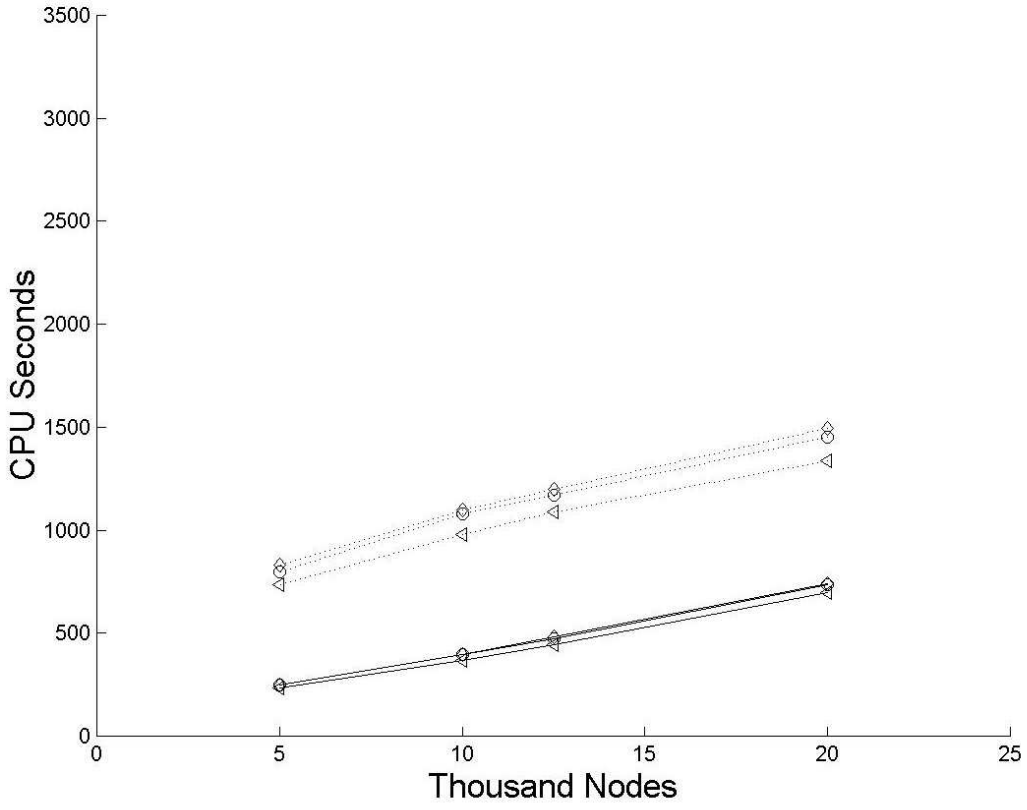

Figure 7. *CPU Seconds for different number of Nodes*

Groups	***G1***	***G2***	***G3***	***G4***	***G5***	***G6***	***G7***	***G8***	***G9***
FL9	11828	7780	6190	8665	5236	4224	7397	4323	4274
FL0	3917	5350	7390	5494	9341	11109	6791	11183	13350
FL1	3408	4675	6476	4797	8312	9853	6000	9910	12364
FL2	4147	5545	7626	5664	9745	11344	6993	11448	13683
FP9	10905	6952	5465	8335	**4945**	**3947**	7240	**4144**	**4036**
FP0	3159	4313	5385	4711	6691	8361	5969	8899	10671
FP1	**2787**	**3780**	**4701**	**4141**	5883	7331	**5220**	7708	9901
FP2	3341	4474	5536	4878	6868	8493	6167	8989	10852

Table 9. *Total Iterations of the Conjugate Gradient*

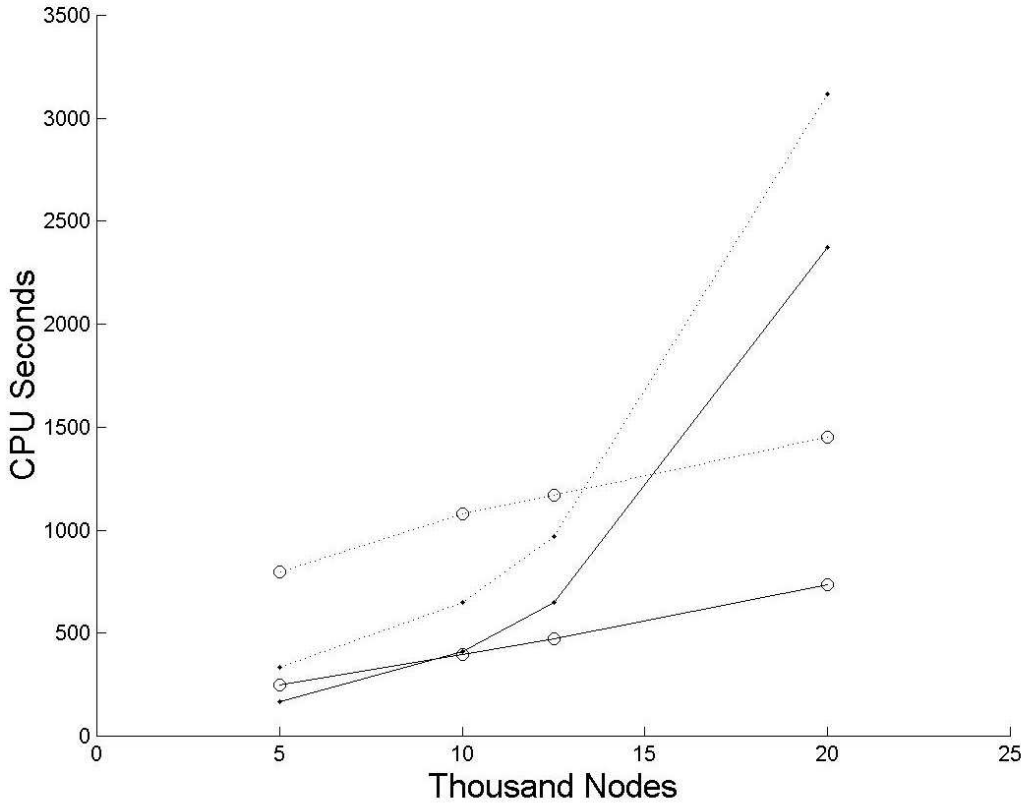

Figure 8. *CPU Seconds for different number of Nodes*

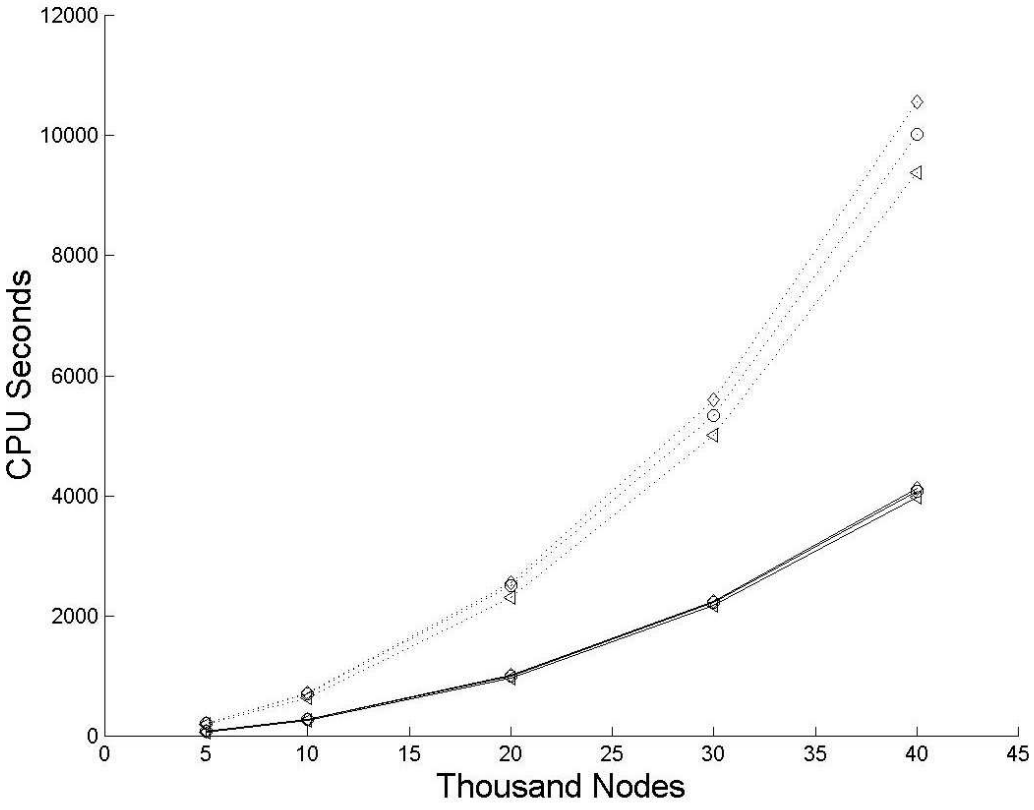

Figure 9. *CPU Seconds for Networks of different sizes*

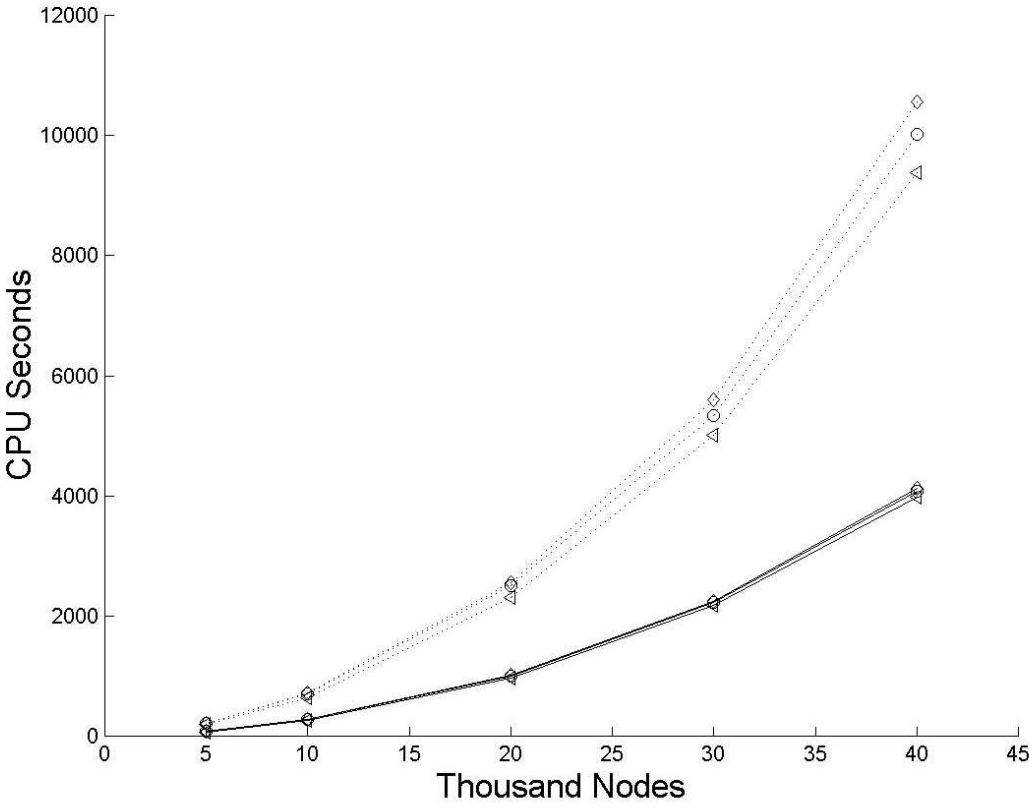

Figure 10. *CPU Seconds for Networks of different sizes*