Proportional Fairness for Combinatorial Optimization

Minh Hieu Nguyen[1], Mourad Baiou[1], Viet Hung Nguyen[1], Thi Quynh Trang Vo[1]

INP Clermont Auvergne, Univ Clermont Auvergne, Mines Saint-Etienne, CNRS, UMR 6158 LIMOS, 1 Rue de la Chebarde, Aubiere Cedex, France
{minh_hieu.nguyen,mourad.baiou,viet_hung.nguyen,thi_quynh_trang.vo}@uca.fr

Mots-clés : *Bi-Objective Combinatorial Optimization, Nash Bargaining Solution, Proportional Fairness, Binary Search Algorithm, Bi-Objective Spanning Tree Problem*

1 Introduction

Proportional fairness (PF) is a concept widely studied in the fields of telecommunications, network design, resource allocation, and social choice. The goal of PF is to provide a compromise between the *utilitarian rule* - which emphasizes overall system efficiency, and the *egalitarian rule* - which emphasizes individual fairness. For example, in wireless communication systems, PF is often used in the allocation of transmission power, bandwidth, and data rates among mobile users to maximize overall system throughput while ensuring fair access for all users [1]. In network scheduling and traffic management, PF plays a significant role in packet scheduling algorithms. It helps ensure that packets from different flows or users are treated fairly, preventing any single user from dominating network resources [2].

If the feasible set is convex, PF coincides with the *Nash bargaining solution* which always exists and can be obtained by maximizing the product of the objectives or equivalently, by maximizing a logarithmic sum problem [1]. Since solving this problem is computationally expensive, in practice, heuristic algorithms are often employed to find approximate solutions that achieve near-PF in some special scenarios (see, e.g., [2]). In contrast, when dealing with non-convex optimization, the existence of PF is not guaranteed and if it exists, it is also the (unique) Nash bargaining solution [1], [3]. Thus, finding PF under non-convexity is not equivalent to any known optimization problem, making it even more difficult. To address such a case, popular approaches involve considering certain non-convex sets, which are convex after a logarithmic transformation [3]. An alternative approach is to introduce the concept of *local proportional fairness*, which is always achievable, and then analyze its properties [4].

This paper proposes a novel approach for investigating PF within the field of combinatorial optimization where the feasible set is generally finite and non-convex. For this purpose, we consider a general *Max-Max Bi-Objective Combinatorial Optimization* (Max-Max BOCO) problem where its two objectives to be simultaneously maximized take only positive values. Then, we seek to find the solution to achieving PF between two objectives which is referred to as *proportional fair solution* (PF solution). We first show that the PF solution, when exists, can be found by maximizing a suitable linear combination of two objectives. Then, our main contribution lies in presenting an exact algorithm that converges within a logarithmic number of iterations to determine efficiently the PF solution. Finally, we provide computational results conducted on the Bi-Objective Spanning Tree Problem (BOSTP) which is a specific example of Max-Max BOCO.

Notice that for Max-Max BOCO where the linear combination of two objectives can be solved in polynomial time, based on our algorithm, the PF solution can be determined in weakly polynomial time. To the best of our knowledge, this is the first approach in combinatorial optimization where an efficient algorithm has been developed for identifying PF.

The paper is organized as follows. In Section 2, we discuss the characterization of the PF solution for Max-Max BOCO. In Section 3, we propose a binary search algorithm for

determining the PF solution. Computational results on some instances of the BOSTP will be presented in Section 4. Finally, in Section 5, we give some conclusions and future works.

2 Characterization of the PF solution for Max-Max BOCO

Let $P(x), Q(x) > 0$ be the two objectives of Max-Max BOCO and $\mathcal{X}$ be the finite and non-convex set of feasible decision vectors x. Let $(P, Q) = (P(x), Q(x))$ denote the objective values corresponding to $x \in \mathcal{X}$. Throughout this paper, the feasible solutions for Max-Max BOCO will be represented by the pairs of objective values (P, Q) instead of the decision vector solutions. Thus, two feasible solutions having the same values of (P, Q) will be considered equivalent and denoted by the notation "$\equiv$". Let $\mathcal{S}$ represent the set of pairs (P, Q) corresponding to all feasible decision vector solutions. Since $\mathcal{X}$ is finite, $\mathcal{S}$ is also a finite set. For Max-Max BOCO, the PF solution (P^{PF}, Q^{PF}) should be such that, if compared to any other solution (P, Q), the aggregate proportional change is non-positive (see, e.g., [1]). Mathematically, we have

$$\frac{P - P^{PF}}{P^{PF}} + \frac{Q - Q^{PF}}{Q^{PF}} \leq 0 \iff \frac{P}{P^{PF}} + \frac{Q}{Q^{PF}} \leq 2, \ \forall (P, Q) \in \mathcal{S}, \tag{1}$$

We will show that the PF solution, if exists, can be obtained by maximizing a suitable linear combination of P and Q by considering the optimization problem:

$$\mathcal{F}(\alpha) = \max_{(P,Q)\in\mathcal{S}} \ f_\alpha(P, Q) := P + \alpha Q,$$

where $\alpha \geq 0$ is an appropriate coefficient to be determined. Notice that we assume the existence of the algorithms for solving $\mathcal{F}(\alpha)$ with $\alpha \geq 0$.

Theorem 1. *$(P^{PF}, Q^{PF}) \in \mathcal{S}$ is the PF solution if and only if it is a solution of $\mathcal{F}(\alpha^{PF})$ with $\alpha^{PF} = P^{PF}/Q^{PF}$.*

Proof. $\implies$ Let (P^{PF}, Q^{PF}) be the PF solution and $\alpha^{PF} = P^{PF}/Q^{PF}$. We have

$$\frac{P}{P^{PF}} + \frac{Q}{Q^{PF}} \leq 2, \ \forall (P, Q) \in \mathcal{S}, \tag{2}$$

Multiplying (2) by $P^{PF} > 0$ and replacing P^{PF}/Q^{PF} by α^{PF}, we obtain

$$P^{PF} + \alpha^{PF} Q^{PF} \geq P + \alpha^{PF} Q, \ \forall (P, Q) \in \mathcal{S},$$

Hence, (P^{PF}, Q^{PF}) is a solution of $\mathcal{F}(\alpha^{PF})$.
$\impliedby$ Let (P^{PF}, Q^{PF}) be a solution of $\mathcal{F}(\alpha^{PF})$ with $\alpha^{PF} = P^{PF}/Q^{PF}$. We have

$$P + \alpha^{PF} Q \leq P^{PF} + \alpha^{PF} Q^{PF}, \ \forall (P, Q) \in \mathcal{S},$$

Replacing α^{PF} by P^{PF}/Q^{PF}, we obtain (2) which implies (P^{PF}, Q^{PF}) is the PF solution. □

Then, the main question now is how to propose a binary search algorithm for determining the PF solution based on Theorem 1. We will address this question in the next section.

3 Binary search algorithm for determining the PF solution

3.1 Algorithm construction

For a given $\alpha_k \geq 0$, let $T(\alpha_k) := P_k - \alpha_k Q_k$ where (P_k, Q_k) is a solution of $\mathcal{F}(\alpha_k)$. According to Theorem 1, let (P^{PF}, Q^{PF}) and α^{PF} denote respectively the PF solution and the PF coefficient such that (P^{PF}, Q^{PF}) is a solution of $\mathcal{F}(\alpha^{PF})$ and $T(\alpha^{PF}) = P^{PF} - \alpha^{PF} Q^{PF} = 0$.

We first show the monotonic relationship between $\alpha \geq 0$ and the solution of $\mathcal{F}(\alpha)$ with respect to the values of P and Q. As a consequence, we also deduce the monotonic relationship between α and $T(\alpha)$.

Lemma 1. *Given $0 \leq \alpha' < \alpha''$ and let (P', Q'), $(P'', Q'') \in \mathcal{S}$ be respectively the solutions of $\mathcal{F}(\alpha')$ and $\mathcal{F}(\alpha'')$. Then $P' \geq P''$ and $Q' \leq Q''$. Moreover, $T(\alpha') > T(\alpha'')$.*

Proof. The optimality of (P', Q') and (P'', Q'') gives

$$P' + \alpha' Q' \geq P'' + \alpha' Q'', \text{ and} \tag{3a}$$
$$P'' + \alpha'' Q'' \geq P' + \alpha'' Q' \tag{3b}$$

Adding (3a) and (3b) gives $(\alpha' - \alpha'')(Q' - Q'') \geq 0$. Since $\alpha' < \alpha''$, $Q' \leq Q''$.
On the other hand, the inequality (3a) implies $P' - P'' \geq \alpha'(Q'' - Q') \geq 0$.
Since $P' \geq P''$, $Q' \leq Q''$ and $\alpha' < \alpha''$, we obtain $T(\alpha') = P' - \alpha' Q' \geq P'' - \alpha' Q'' > P'' - \alpha'' Q'' = T(\alpha'')$. □

According to Lemma 1, if $\alpha' < \alpha''$ and (P', Q') is the solution of both $\mathcal{F}(\alpha')$ and $\mathcal{F}(\alpha'')$ then (P', Q') is the solution of $\mathcal{F}(\alpha)$, $\forall \alpha \in (\alpha', \alpha'')$. For given $0 \leq \alpha_i < \alpha_j$ and $T(\alpha_i)T(\alpha_j) > 0$, we also have $\alpha^{PF} \notin (\alpha_i, \alpha_j)$ because for $\alpha' \in (\alpha_i, \alpha_j)$ and an arbitrary solution (P', Q') of $\mathcal{F}(\alpha')$, $T(\alpha')$ has the same sign as $T(\alpha_i)$ and $T(\alpha_j)$ which implies $T(\alpha') \neq 0$ and then $\alpha' \neq \alpha^{PF}$.

Let α^{sup} be an upper bound of α^{PF} such that $\alpha^{PF} < \alpha^{sup}$. According to the results of Theorem 1 and Lemma 1, the main idea of our algorithm is based on a binary search algorithm in the interval $[0, \alpha^{sup}]$. More precisely, we use Procedure *SEARCH*() to identify the PF solution and the PF coefficient α^{PF} in such interval, ensuring that $T(\alpha^{PF}) = 0$. Starting from an interval $[\alpha_i, \alpha_j] \subseteq [0, \alpha^{sup}]$ with $T(\alpha_i) > 0$ and $T(\alpha_j) < 0$, Procedure *SEARCH*() selects α_s as the midpoint of the interval $[\alpha_i, \alpha_j]$ and solve $\mathcal{F}(\alpha_s)$ to obtain a solution (P_s, Q_s). Then, we use Procedure *Verify_PF_sol*(α_s, P_s, Q_s) to verify whether (P_s, Q_s) is the PF solution. If the verification is unsuccessful, the half-interval in which the PF coefficient cannot exist is eliminated and we retain only one half-interval for further exploration within Procedure *SEARCH*(). The choice is made between $[\alpha_i, \alpha_s]$ and $[\alpha_s, \alpha_j]$, depending on the sign of $T(\alpha_s)$. The parameter ϵ guarantees the absence of the PF coefficient in an interval whose length is less than ϵ. In other words, for an interval $[\alpha_l, \alpha_l + \epsilon]$, there exists a feasible solution $(P_l, Q_l) \not\equiv (P^{PF}, Q^{PF})$ which is a solution of both $\mathcal{F}(\alpha_l)$ and $\mathcal{F}(\alpha_l + \epsilon)$. Thus, our algorithm always converges in a logarithmic number of iterations in terms of ϵ and α^{sup}. Note that we can set ϵ to a predefined small positive value (e.g. 0.01).

3.2 Algorithm statement and proofs

In this section, we first introduce Procedure *Verify_PF_sol*(α_0, P_0, Q_0) to verify whether a solution (P_0, Q_0) of $\mathcal{F}(\alpha_0)$ is the PF solution. Its proof will be stated in the next lemma.

Procedure 1 Verify whether a solution (P_0, Q_0) of $\mathcal{F}(\alpha_0)$ is the PF solution

Input: $\alpha_0 \geq 0$, $(P_0, Q_0) \in \mathcal{S}$ is a solution of $\mathcal{F}(\alpha_0)$.
Output: True if (P_0, Q_0) is the PF solution or False otherwise.
1: **procedure** VERIFY_PF_SOL(α_0, P_0, Q_0)
2: $\alpha' \leftarrow P_0/Q_0$
3: solving $\mathcal{F}(\alpha')$ to obtain the solution (P', Q').
4: **if** $f_{\alpha'}(P', Q') = f_{\alpha'}(P_0, Q_0)$ **then** return True
5: **else** return False
6: **end if**
7: **end procedure**

Lemma 2. *Given $\alpha_0 \geq 0$ and $(P_0, Q_0) \in \mathcal{S}$ as a solution of $\mathcal{F}(\alpha_0)$. Let $\alpha' = P_0/Q_0$ and (P', Q') be a solution of $\mathcal{F}(\alpha')$. Then (P_0, Q_0) is the PF solution if and only if $f_{\alpha'}(P', Q') = f_{\alpha'}(P_0, Q_0)$.*

Proof. $\Longrightarrow$ If (P_0, Q_0) is the PF solution then (P_0, Q_0) is also a solution of $\mathcal{F}(\alpha')$ due to Theorem 1. Thus, $f_{\alpha'}(P', Q') = f_{\alpha'}(P_0, Q_0)$.

$\Longleftarrow$ If $f_{\alpha'}(P', Q') = f_{\alpha'}(P_0, Q_0)$ then (P_0, Q_0) is also a solution of $\mathcal{F}(\alpha')$. Since $\alpha' = P_0/Q_0$, (P_0, Q_0) is the PF solution due to Theorem 1. □

Subsequently, from $0 \leq \alpha_i < \alpha_j$ and (P_i, Q_i), (P_j, Q_j) as the solutions of $\mathcal{F}(\alpha_i)$ and $\mathcal{F}(\alpha_j)$, we present Procedure $SEARCH(\alpha_i, P_i, Q_i, \alpha_j, P_j, Q_j, \epsilon)$ for determining the PF solution where the PF coefficient α^{PF} is in the interval $[\alpha_i, \alpha_j]$.

Procedure 2 Determine the PF solution where the PF coefficient is in the interval $[\alpha_i, \alpha_j]$

Input: a positive parameter ϵ small enough, (α_i, P_i, Q_i) and (α_j, P_j, Q_j) satisfy:

- $0 \leq \alpha_i < \alpha_j$, $(P_i, Q_i), (P_j, Q_j)$ are respectively solutions of $\mathcal{F}(\alpha_i)$ and $\mathcal{F}(\alpha_j)$.
- (P_i, Q_i) and (P_j, Q_j) are not PF solutions, $T(\alpha_i) > 0$ and $T(\alpha_j) < 0$.

Output: The PF solution if it exists or Null otherwise.

procedure $SEARCH(\alpha_i, P_i, Q_i, \alpha_j, P_j, Q_j, \epsilon)$
  if $\alpha_j - \alpha_i \geq \epsilon$ **and** $(P_i, Q_i) \not\equiv (P_j, Q_j)$ **then**
    $\alpha_s \leftarrow (\alpha_i + \alpha_j)/2$
    solving $\mathcal{F}(\alpha_s)$ to obtain a solution (P_s, Q_s)
    if *Verify_PF_sol*(α_s, P_s, Q_s) == True **then** return (P_s, Q_s)
    end if
    $T(\alpha_s) \leftarrow P_s - \alpha_s Q_s$
    if $T(\alpha_s) > 0$ **then** return $SEARCH(\alpha_s, P_s, Q_s, \alpha_j, P_j, Q_j, \epsilon)$
    else if $T(\alpha_s) < 0$ **then** return $SEARCH(\alpha_i, P_i, Q_i, \alpha_s, P_s, Q_s, \epsilon)$
    end if
  else return Null
  end if
end procedure

Finally, our algorithm to determine the PF solution can be stated as follows.

Algorithm 3 Determine the PF solution for Max-Max BOCO

Input: An instance of Max-Max BOCO, a positive parameter ϵ small enough.
Output: PF solution if it exists or Null otherwise.

solving $\mathcal{F}(0)$ to obtain a solution (P_0, Q_0)
if $Verify\_PF\_sol(0, P_0, Q_0)$ == True **then** return (P_0, Q_0)
end if
$\alpha^{sup} \leftarrow P_0/Q_0 + 1$
solving $\mathcal{F}(\alpha^{sup})$ to obtain a solution (P^{sup}, Q^{sup})
if $Verify\_PF\_sol(\alpha^{sup}, P^{sup}, Q^{sup})$ == True **then** return (P^{sup}, Q^{sup})
else return $SEARCH(0, P_0, Q_0, \alpha^{sup}, P^{sup}, Q^{sup}, \epsilon)$
end if

Notice that since $0 < \alpha^{PF}$ and $0 < \alpha^{sup}$, $P_0 \geq P^{PF}, Q_0 \leq Q^{PF}$ and $P_0 \geq P^{sup}, Q_0 \leq Q^{sup}$ due to Lemma 1. Thus, $\alpha^{PF} = P^{PF}/Q^{PF} \leq P_0/Q_0 < \alpha^{sup}$. Moreover, $T(0) = P_0 > 0$ and $T(\alpha^{sup}) = P^{sup} - \alpha^{sup} Q^{sup} < P^{sup} - \frac{P_0}{Q_0} Q^{sup} \leq 0$.

By the following theorem, we will show that Algorithm 3 can determine the PF solution in a logarithmic number of iterations in terms of ϵ and α^{sup}.

Theorem 2. *Algorithm 3 can determine the PF solution in a logarithmic number of iterations in terms of ϵ and α^{sup}.*

Proof. The execution of Algorithm 3 is based on the binary search algorithm for the interval $[0, \alpha^{sup}]$ with a length equals α^{sup}. At each iteration, we divided an interval into two half-intervals with equal length. Then, the half in which the PF coefficient cannot exist is eliminated

and the search continues on the remaining half. Since Algorithm 3 terminated in the worst case when it found an interval with a length smaller than ϵ, the number of iterations for Algorithm 3 is $O(\log_2 \frac{\alpha^{sup}}{\epsilon})$. Consequently, Algorithm 3 can determine the PF solution in a logarithmic number of iterations in terms of ϵ and α^{sup}. □

As a result of Theorem 2, if solving $\mathcal{F}(\alpha)$ can be done in polynomial time, the PF solution can be determined in weakly polynomial time. Notice that the Bi-Objective Spanning Tree Problem (BOSTP) considered in the next section belongs to this category.

4 Experimental study on the BOSTP

4.1 Definition, modeling and algorithm for the BOSTP

In this section, we consider a BOSTP which merges the Maximum STP, which involves maximizing the total profit, and the Max-Min STP, where the goal is to maximize the minimum edge reliability. For the BOSTP, we find a spanning tree achieving PF between two objectives: the total profit representing the overall efficiency and the minimum of the edge reliability representing the individual fairness. For the simplicity of calculation, we suppose that the values of profit and reliability are positive integers. Furthermore, we set the value of ϵ as 0.01.

We consider a finite, connected, undirected graph $G = (V, E)$ where $V = [n] := \{1, ..., n\}$ with $n \geq 2$, $|E| = m$ and $p_e, r_e \in \mathbb{Z}_+$ are two weights associated with edge $e \in E$ representing respectively profit and reliability on this edge. Let $\mathcal{T}(G)$ denote the set of all spanning trees in G. Let $P, Q > 0$ denote respectively the total profit and the minimum edge reliability in a spanning tree T of G. We have $P = \sum_{e \in T, T \in \mathcal{T}(G)} p_e$ and $Q = \min_{e \in T, T \in \mathcal{T}(G)} r_e$.

When $\alpha = 0$, solving $\mathcal{F}(\alpha)$ is equivalent to solving the Maximum STP which can be done in polynomial time by using Kruskal's algorithm [5]. For solving $\mathcal{F}(\alpha)$ with $\alpha > 0$, we construct a polynomial-time algorithm which is similar to the one for solving Min-Max STP where the maximum edge weight is to be minimized [6]. It is based on there are at most m different values of Q. Let $r_1 \leq r_2 \leq ... \leq r_m$ be the sorted list of reliability values corresponding to m edges $e_1, e_2, ..., e_m$ and let $E^i = \{e_j | r_j \geq r_i\}, \forall i = 1, 2, .., m$. We select respectively the minimum edge reliability as r_i for $i = 1, 2, ..., m$ and then we determine a spanning tree whose edges are in E^i that maximizes the sum of profit and contains the edge e_i (i.e., we solve the Maximum STP on the edge set E^i with the fixed edge e_i). Notice that E^i may not contain a feasible spanning tree (in this case, we set the optimal value of $f_\alpha(P, Q)$ as 0). By comparing the corresponding optimal values of $f_\alpha(P, Q)$ in these m selections, we obtain a solution for $\mathcal{F}(\alpha)$. Hence, the PF solution for the BOSTP can be determined in weakly polynomial time.

4.2 Computational results on the instances of the BOSTP

We investigate the performance of the presented algorithm for the BOSTP on the random NetworkX graph. It returns a $G_{n,pro}$ random graph, also known as an Erdos-Renyi graph where n is the number of nodes and pro is the probability for edge creation. For this paper, the number of nodes is selected from the interval [30,40] with probability $pro = 0.5$. Moreover, the edge profit and the edge reliability are generated uniformly randomly respectively in the intervals $[100, 900]$ and $[10, 90]$. The optimal solutions for the Maximum STP, Max-Min STP, and BOSTP are shown in Table 1 where the values of P, Q in case the PF solution does not exist are denoted as "Null". Note that "GNn" represents a generated graph with n nodes. All the experiments are conducted on a PC Intel Core i5-9500 3.00GHz with 6 cores and 6 threads.

According to Table 1, we obtained the PF solutions for most instances and they are different from the solutions of the Maximum STP and the Max-Min STP. Generally, the PF solutions offer a more favorable compromise between two objectives than the solutions of the Maximum STP (resp. Max-Min STP): the significant increase in the values of Q (resp. P) compared to the slight drop in the values of P (resp. Q) in percentage. Table 1 also indicates that our

TAB. 1: Computational results of Maximum STP, Max-Min STP and BOSTP

Instance	Maximum STP			Max-Min STP			BOSTP			
	P	**Q**	**Time**	**P**	**Q**	**Time**	**P**	**Q**	**Time**	**Iters**
GN30	24259	12	0.10	14062	74	0.30	21633	67	3.28	2
GN32	24272	16	0.07	13359	74	0.08	18651	71	1.96	2
GN34	28329	11	0.08	19314	77	0.24	Null	Null	5.24	5
GN36	28554	17	0.05	17944	69	0.25	25138	68	4.42	2
GN38	33531	10	0.14	20432	73	0.24	28358	72	6.45	3
GN40	33681	12	0.14	17789	79	0.65	29171	68	5.07	2

algorithm seems to converge quickly in terms of time calculation and number of iterations, especially when the PF solution exists. Another important remark is that the existence of the PF solution seems to be much related to the edge weights and the structure of the graph rather than to the size of the graph. Although we selected randomly the values of profit and reliability, they appeared with a high frequency, approximately 85% over the total tested instances.

5 Conclusion

In this paper, we have utilized *proportional fairness* in the context of Max-Max BOCO where the two objectives to be maximized take only positive values and the feasible set is finite and non-convex. We considered a general Max-Max BOCO problem where we looked for a solution achieving proportional fairness between two objectives - which is referred to as PF solution. We first presented the characterization of the PF solution for Max-Max BOCO. Then, we designed an exact algorithm that converges within a logarithmic number of iterations to determine the PF solution. Computational experiments on some instances of the Bi-Objective Spanning Tree Problem have shown the efficiency of our algorithm, indicating its rapid convergence.

For future works, in cases the PF solution does not exist, we are interested in modifying our algorithm to provide a near-PF solution maximizing the product of the objectives, resembling a generalized Nash bargaining solution. Furthermore, the results of this paper could be extended to multi-objective combinatorial optimization involving more than two objectives.

References

[1] Kelly, F.P. et al.: Rate control for communication networks: shadow prices, proportional fairness, and stability. In: Journal of the Oper. Res. Society, 49(3), November 1997.

[2] Kushner, H.J. et al.: Convergence of proportional-fair sharing algorithms under general conditions. In: IEEE Transactions on Wireless Communications, 3(4):1250–1259, July 2004.

[3] Boche, H., and Schubert, M.: Nash Bargaining and Proportional Fairness for Wireless Systems. In: IEEE/ACM Transactions on Networking, Vol. 17, No. 5, October 2009.

[4] Brehmer, J., and Utschick, W.: On Proportional Fairness in Non-convex Wireless Systems. In: International ITG Workshop on Smart Antennas - WSA, Berlin, February 2009.

[5] Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. In: American Mathematical Society, 7(1), 1956, pp 48-50.

[6] Camerini, P.M.: The Min-Max Spanning Tree Problem. In: Information Processing Letters, Vol. 7, Number 1, 1978.