Compare NSGA-II, SPEA2, MOEA/D Performance for Task Assignment Problem

1. DESCRIPTION

The aim of these experiments is to explore the multiobejctive algorithms framework (e.g. jMetal, MOEA, or any other) and identify the most effective multi-objective algorithm, which gives an approximation set to the Pareto front of the problem with best convergence and uniform diversity

2. PROBLEM FORMULATION

In this section, we describe our problem formulation which is an extension of the problem defined by Younas et al. [**?**]. The problem discussed in this paper deals with multi-objective optimal assignment of tasks to collaborating teams of agents. Let $\mathcal{A} = \{a_i | i = 1, \ldots, m\}$ be the set of m agents (candidate members) and let $\mathcal{T} = \{t_j | j = 1, \ldots, n\}$ be the set of n tasks, where $m \geq n$. Each agent $a_i \in \mathcal{A}$ has a set of p attributes $c_i = \{c_{ik} | k = 1, \ldots, p\}$, which are real values that quantify its capabilities. Similarly for each task $t_j \in \mathcal{T}$, a weight is associated to each capability and it defines capability weight vector $w_j = \{w_{kj} | k = 1, \ldots, p\}$. Assume that each task t_j requires a team of agents with a fixed number b_j of agents. Each agent $a_i \in \mathcal{A}$ can perform at most one task. We denote the team of b_j agents performing task t_j by $team_{\mathcal{S}_j}$, where the index $\mathcal{S}_j$ is a set

$$\mathcal{S}_j \subset \{1, 2, \ldots m\}, \quad |\mathcal{S}_j| = b_j, \quad a_i \in team_{\mathcal{S}_j} \Leftrightarrow i \in \mathcal{S}_j.$$

We also assume that total completion time for each task t_j is $time_j$ (time units), in our case $time_j$ represent number of hours required to perform task t_j. Assuming each agent $a_i \in \mathcal{S}_j$ participating in $team_j$ works for equal $\left(\frac{time_j}{b_j}\right)$ number of hours on task t_j. Each agent a_i has a salary $Salary_i$, which is a function of its capabilities and certain weights (values) for those capabilities $h(c_{ik}, v_k | k = 1, \ldots, p)$, where v_k defines relative value for capability k for calculating salary. Within each team, agents collaborate with each other which improve the quality of the task performed. The execution of each task also incurs some cost which depends on completion time and salary of the agents assigned to the task.

GECCO'13, July 6-10, 2013, Amsterdam, The Netherlands.

The goal is to optimally assign all tasks to teams of agents such that quality is maximized and cost is minimized. Now we proceed to models for calculating these two objectives, the quality and the cost.

2.1 Model for quality calculation

We assume that agents participating in a team collaborate within team environment and they produce value (quality) which is a non-linear function of these agents and the assigned task. Thus $team_{\mathcal{S}_j}$, which performs task t_j produces the quality $f(team_{\mathcal{S}_j}, t_j)$. In order to calculate the quality, we consider the team-working model introduced in [**?**] which is an extension of the base model introduced in [**?**] and [**?**]. The overall quality of performing all tasks is defined as sum of values produced by the teams. One of the objectives is to maximize the quality i.e., maximizing the objective function

$$\mathcal{Z}_1(\mathcal{S}_1, \ldots \mathcal{S}_n) = \sum_{j=1}^{n} f(team_{\mathcal{S}_j}, t_j). \tag{1}$$

An agent in a team may be influenced by the capabilities of other participating agents. The quality produced by a team is assumed to be equal to the sum of values produced by the participating agents:

$$f(team_{\mathcal{S}_j}, t_j) = \sum_{i \in \mathcal{S}_j} q(a_i, team_{\mathcal{S}_j}, t_j). \tag{2}$$

If a task is performed by a single agent, then quality produced by the agent is equal to weighted sum of the agents capabilities, i.e.,

$$q(a_i, t_j) = \sum_{k=1}^{p} c_{ik} w_{kj}. \tag{3}$$

However, if a task is performed by a team, the agents are influenced by other team members and capability of each type is influenced by maximum capability ($c_k^{\max}$) of that type. The new capabilities are calculated as

$$c'_{ik} = c_{ik} + \frac{c_{ik}(c_k^{\max} - c_{ik})}{c_k^{\max}}, \tag{4}$$

where $c_k^{\max} = \max_{i \in \mathcal{S}_j} \{c_{ik}\}, \quad \forall k \in \{1, \ldots, p\}$.

Equation (4) indicates the following:

1. A capability c_{ik} of an agent a_i improves only when its value is less than the maximum capability $c_k^{\max}$ of that type in a team. Thus the agents with such capabilities benefit from the team working.

2. A capability c_{ik} of an agent a_i where $c_{ik} = c_k^{\max}$ is not affected by collaboration.

3. A capability $c_{ik} = 0$ of an agent a_i is not affected by team collaboration.

4. A capability c_{ik} of an agent a_i where $c_{ik} = \frac{c_k^{\max}}{2}$ has the highest improvement due to collaboration.

From equation (3) and (4), the quality achieved by an agent while working and collaborating with other team members is obtained as

$$q(a_i, team_{\mathcal{S}_j}, t_j) = \sum_{k=1}^{p}(c_{ik} + \frac{c_{ik}(c_k^{\max} - c_{ik})}{c_k^{\max}})w_{kj}. \quad (5)$$

By substituting (5) and (2) into the objective function (1), we get the following final quality function:

$$\mathcal{Z}_1 = \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{k=1}^{p}(c_{ik} + \frac{c_{ik}(c_k^{\max} - c_{ik})}{c_k^{\max}})w_{kj}x_{ij}, \quad (6)$$

where $c_k^{\max} = \max_{1 \le i \le m}\{c_{ik}x_{ij}\}$
and $x_{ij} = 1$ if task t_j is assigned to agent a_i and 0 otherwise.

2.2 Model for cost calculation

The cost of performing a given task t_j depends on the capabilities ($c_l = \{c_{l1}, \ldots, c_{lp}\}, \quad \forall l \in \mathcal{S}_j$) of assigned agents and the estimated completion time $time_j$. Each agent a_i has a salary $Salary_i$, which is a function of its capabilities and certain weights (values) for those capabilities $h(c_{ik}, v_k|k = 1, \ldots, p)$, where v_k defines relative value for capability k for calculating salary.

The overall cost is defined as the sum of the costs of performing all given tasks. Our objectives is to minimize the cost i.e., minimizing the objective function

$$\mathcal{Z}_2(\mathcal{S}_1, \ldots \mathcal{S}_n) = \sum_{j=1}^{n} g(team_{\mathcal{S}_j}, t_j). \quad (7)$$

Here g is the cost function

$$g(team_{\mathcal{S}_j}, t_j) = \sum_{i \in \mathcal{S}_j} cost(a_i, team_{\mathcal{S}_j}, t_j). \quad (8)$$

Moreover, we assume that all assigned agents to a task work for the same duration (equal number of hours). The cost of a single agent a_i assigned to a task t_j is calculated as:

$$cost(a_i, team_{\mathcal{S}_j}, t_j) = Salary_i(\frac{time_j}{|\mathcal{S}_j|}). \quad (9)$$

Salary of an agent a_i is defined by

$$Salary_i = h(c_{ik}, v_k), \forall k \in \{1, \ldots, p\}, \quad (10)$$

where v_k defines relative value for capability k. We use a simple model here and estimate the salary as the weighted sum of the capabilities, i.e. $h(c_{ik}, v_k) = \sum_{k=1}^{p} c_{ik}v_k$.

By substituting (10) and (8) into the objective function (7), we get the following final cost function:

$$\mathcal{Z}_2 = \sum_{j=1}^{n}(\frac{time_j}{|\mathcal{S}_j|}\sum_{i=1}^{m}\sum_{k=1}^{p} c_{ik}v_k x_{ij}), \quad (11)$$

where $x_{ij} = 1$ if task t_j is assigned to agent a_i and 0 otherwise.

2.3 A Multi-Objective Model with Constraints

In this section, we present a bi-objective model with constraints. Our goal is to maximize quality and minimize cost while assigning all given tasks to collaborating teams of agents. Mathematical formulation is summarized as follows:

$$Maximize \quad \mathcal{Z}_1 = \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{k=1}^{p}(c_{ik} + \frac{c_{ik}(c_k^{\max} - c_{ik})}{c_k^{\max}})w_{kj}x_{ij}, \quad (12)$$

$$Minimize \quad \mathcal{Z}_2 = \sum_{j=1}^{n}(\frac{time_j}{|\mathcal{S}_j|}\sum_{i=1}^{m}\sum_{k=1}^{p} c_{ik}v_k x_{ij}), \quad (13)$$

subject to constraints

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall a_i \in \mathcal{A} \quad (14)$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \qquad \forall t_j \in \mathcal{T} \quad (15)$$

$$b_j \ge 1, \qquad \forall t_j \in \mathcal{T} \quad (16)$$

$$\sum_{j=1}^{n} b_j \le m \quad (17)$$

$$x_{ij} \in \{0, 1\}, \qquad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T} \quad (18)$$

where $b_j = |\mathcal{S}_j|$, $\mathcal{S}_j$ is a set of assigned agents to task t_j. Constraint (14) states that an agent can perform at most one task. Constraint (15) ensures that each task t_j is performed by a $team_{\mathcal{S}_j}$, which consists of b_j agents. Constraint (16) means that each $team_{\mathcal{S}_j}$ should have at least one agent and constraint (17) ensures that number of agents assigned to all n teams should be equal or less than total number of available agents m. The last constraint (18) is an assignment matrix, where $x_{ij} = 1$ means task t_j is assigned to agent a_i.

3. PROBLEM INSTANCES (DATA SET)

In order to make the analysis and discussion unbiased, we consider more than one instances of collaborating teams assignments with different problem sizes as shown in Table 1. In all problem instances, each agent a_i has 10 different capabilities $c_i = \{c_{ik}|k = 1, \ldots, 10\}$ and each task t_j has certain requirements (weights) $w_j = \{w_{kj}|k = 1, \ldots, 10\}$ for these capabilities. We assume that the values of c_{ik} and w_{kj} are integer values between 0 and 4 inclusive, where 0 is the lowest and 4 is the highest value for the agent's capability and the capability's weight for the task. Assuming that most of the agents have average capability values and most of the tasks require average capabilities, we generate input data with smaller probabilities to extreme values and higher probabilities to average values. The synthetic data is generated randomly such that $p(c_{ik} = 4.0) = 0.1, p(c_{ik} = 3.0) = 0.15, p(c_{ik} = 2.0) = 0.5, p(c_{ik} = 1.0) = 0.15$ and $p(c_{ik} = 0.0) = 0.1$. The same distribution is followed while generating weights w_{kj} for the task t_j. We also assume that

Table 1: Specification of the Collaborating Teams Assignment Problem Instances

Prob#	Total Agents	Required Agents	Number of Tasks	Teams (# of agents assigned to the tasks)
1	10	10	4	[2, 3, 2, 3]
2	50	20	4	[3, 4, 6, 7]
3	400	400	100	100 teams (each team has 4 agents)

total completion time for each task t_j is $time_j$ (time units). In our case $time_j$ represents number of hours required to perform task t_j, which is generated with uniform distribution on $[1000, 1500]$. It is assumed that each agent $a_i \in \mathcal{S}_j$, participating in $team_j$ works for equal $\left(\frac{time_j}{b_j}\right)$ number of hours on task t_j. Each agent a_i has a salary $Salary_i$, which is a function of its capabilities and certain weights (values) for those capabilities $h(c_{ik}, v_k | k = 1, \ldots, p)$, where v_k defines relative value for each capability k for calculating salary and we assume $v_k = 1$ for each capability. The execution of each task also incurs some cost, which depends on completion time and salary of the resources (agents) assigned to the task.

3.1 Performance comparison

To compare and discuss the performance of NSGA-II SPEA2, and MOEA/D a set of computational experiments are performed with different problem sizes as shown in Table 1. NSGA-II SPEA2, and MOEA/D are stochastic in nature and different replications can produce different results. In order to overcome this problem, we consider 10 independent replications for each algorithm and compare the results by considering mean and standard deviation. In evolutionary algorithms, initial population size play an important role in the quality of the results. Different algorithms show different behavior on different population sizes. Considering constant number of ïňĄtness evaluations, population size can play an important role in evolution. There is some trade-off as far as size of the population is concerned. For example if we assume small population size, then sometimes less number of evaluations may be required for convergence of the population to the Pareto-optimal solutions. But sometimes smaller population can be cause of pre-mature termination (algorithm can be trapped by local minima/maxima). To make the discussion fair, for each problem size, we conduct 10 replications of each algorithm on ïňĄve different initial population sizes (20, 40, 80, 160 and 320) and each algorithm terminates after 100, 000 function evaluations. For each replication, we calculate the Hypervolume, Spread and GD, and IGD of the obtained Pareto front.

Calculate mean and standard deviation of Hypervolume, Spread and GD, and IGD of 10 replications for each population size.

Example Plot: Draw plots for Hypervolume, Spread and Coverage as shown in Figure 1. You also have to include curve for MOEA/D.

Example Table: You also need to have tables for mean and standard deviation of Hypervolume, Spread and GD, and IGD as shown in Table 2.

Reference Pareto front: For each problem, we integrate all obtained nondominated set of solutions for all 10 replications on 20, 40, 80, 160 and 320 initial population sizes.

Assignment should be done in jMetal5.3 or MOEA Framework or any other available framework, the algorithms are already implemented, you have to explore the framework and perform the experiments. You need to submit plots with results in tables and the source files.

Table 2: Hypervolume Results: (on initial population size = 160)

Prob#	NSGA-II		SPEA2	
	$\overline{HV}$	σ_{HV}	$\overline{HV}$	σ_{HV}
1	0.7986	0.0	0.7986	0.0
2	0.8412	0.0	0.8407	0.0032
3	0.5917	0.0897	0.5176	0.0974

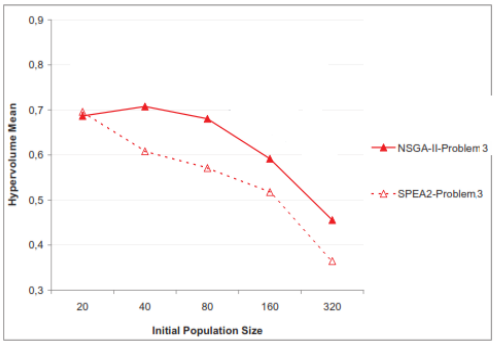

Figure 1 : NSGA-II and SPEA2 Comparison

Figure 1: NSGA-II and SPEA2 Comparison