ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-246

Ideal Point Guided Iterative Deepening

J. Coego and **L. Mandow** and **J.L. Pérez de la Cruz** [1]

Abstract. Many real world search problems involve different objectives, usually in conflict. In these cases the cost of a transition is given by a cost vector. This paper presents IPID, a new exact algorithm based on iterative deepening, that finds the set of all Pareto-optimal paths for a search problem in a graph with vectorial costs. Formal proofs of the admissibility of IPID are presented, as well as the results of some empirical comparisons between IPID and other approaches based on iterative deepening. Empirical results show that IPID is usually faster than those approaches.

1 Introduction

Multiobjective problems arise in many different real world domains, from combinatorial auctions [8] to resource allocation problems [9], from channel routing [2] to domain independent planning [7]. Exact multiobjective search algorithms do not deal with a single optimal solution cost as single objective algorithms do, but with a set of several optimal costs with traded-off objectives. For example, route planning in road maps [6], a current research topic among transportation problems, involves two relatively uncorrelated objectives: economic cost (including fuel and tolls) and travel time. The main goal of multiobjective search is to minimize both components. However, decreasing travel time may cause obvious increasing costs because of larger fuel consumption and possibly tolls.

Exact multiobjective approaches try to locate the whole set of optimal solution costs (also known as Pareto-optimal solutions or non-dominated solutions). Both best-first and depth-first algorithms have been designed for this task; however, in tree-shaped search spaces depth-first algorithms are the natural choice, since —contrary to best-first algorithms— they present worst-case linear-space complexity.

This paper presents Ideal Point Iterative Deepening (IPID), a new exact depth-first multiobjective algorithm which extends the single-objective IDA* algorithm [4]. Previous approaches to this extension have been proposed: IDMOA* [3] and PIDMOA* [1]. IPID aims to improve the performance of both algorithms by dealing with their respective weaknesses: unnecessary reexpansion of nodes in IDMOA* and excessive number of dominance tests in PIDMOA*. In fact, experimental tests presented in this paper show reductions of execution time by a factor of 3 over these algorithms.

The structure of this paper is as follows. Section 2 presents the basics of multiobjective search. Section 3 describes IPID, the new algorithm, presented through both its explained pseudocode and an example of its execution. It also includes formal proofs on the admissibility of IPID. Section 4 presents the experimental testbed used in order to compare IPID with previous proposals and discusses the obtained results, mainly in terms of time consumption. Finally, in section 5 some conclusions and future work are outlined.

[1] Dpto. Lenguajes y Ciencias de la Computación. Universidad de Málaga, Spain, email: {jcoego,lawrence,perez}@lcc.uma.es

2 Basics of Multiobjective Search Problems

This section summarizes the main concepts used in multiobjective search that are needed to define algorithm IPID. In multiobjective problems, the cost of reaching a node is represented by a q-dimensional vector cost $\vec{g} = \{o_1, \ldots, o_q\}$, where q is the number of objectives being considered. Given two cost vectors $\vec{g_1}$ and $\vec{g_2}$, we define the *dominance* relation ($\prec$) as $\vec{g_1} \prec \vec{g_2}$ iff $\forall i (1 \leq i \leq q)$ $\vec{g_1}(i) \leq \vec{g_2}(i)$ and $\vec{g_1} \neq \vec{g_2}$. The dominance relation implies that the *dominating* vector has at least a cost component smaller than the *dominated* vector. Similarly a weaker relation, *"dominates or equals"* ($\preceq$) is defined as $\vec{g_1} \preceq \vec{g_2}$ iff $\forall i (1 \leq i \leq q)$ $\vec{g_1}(i) \leq \vec{g_2}(i)$. Also an *indifference* relation ($\sim$) is defined as $\vec{g_1} \sim \vec{g_2}$ iff neither $\vec{g_1}$ dominates $\vec{g_2}$ nor $\vec{g_1}$ dominates $\vec{g_2}$. Finally we define the relation *strictly-better* ($\ll$) as $\vec{g_1} \ll \vec{g_2}$ iff $\forall i (1 \leq i \leq q)$ $\vec{g_1}(i) < \vec{g_2}(i)$.

Given a set of vectors T, $nodomset(T) = \{\vec{t} \in T \mid \nexists \vec{u} \in T \quad \vec{u} \prec \vec{t}\}$. The elements of $nodomset(T)$ are called *nondominated* or *Pareto-optimal*. The ideal point $i\vec{Point}$ of T is given by the best components $i\vec{Point}(i)$ that can be found in any vector in T, i. e., $\forall i, 1 \leq i \leq q$, $i\vec{Point}(i) = \min\{\vec{v}(i) \mid \vec{v} \in T\}$.

Multiobjective search performs an exploration of a graph G with a set N of nodes and a set A of arcs connecting pairs of nodes. We assume a start node $s \in N$ and a set of goal nodes $\Gamma \subseteq N$. We assume that $H(n)$ returns a nondominated set of estimated costs $\vec{h}(n)$ from n to Γ. A multiobjective heuristic function $H(n)$ is admissible when for all non-dominated solution paths P^* = (s, n_1, ...,n_i,n_{i+1}, ...,γ_k), $\gamma_k \in \Gamma$ and each subpath P_i^* = (s,n_1, ...,n_i) of P^*, there exists $\vec{h} \in H(n_i)$ such that $\vec{g}(P_i^*) + \vec{h} \preceq \vec{g}(P^*)$. $F(n)$ is the set of all the $\vec{f}(n)$ vectors such that there are $\vec{g}(n), \vec{h}(n)$ with $\vec{f}(n) = \vec{g}(n) + \vec{h}(n)$.

Given G, s and Γ, a multiobjective search algorithm is admissible iff it returns exactly the whole set of Pareto-optimal solutions. This set is also referred to as C^*.

3 Algorithm IPID

Ideal Point Iterative Deepening (IPID) is an algorithm which extends the notion of heuristic iterative deepening search presented in [4] (IDA*) to the multiobjective case. Iterative deepening search performs consecutive depth-first searches, each one bounded by a cost value computed in the previous iteration. When the cost of the node being explored exceeds the current bound, then search is discontinued. In IDA* this bound is computed by taking the minimum f value of the nodes discarded in the previous iteration. IDA* can be proven to be admissible under reasonable assumptions.

Extending this idea to the multiobjective case is not straightforward, since vectorial costs are now involved. Computing the non-dominated values in a set of cost vectors will usually give a *set* of bounds, due to the partial order nature of the dominance relation.

Two different multiobjective extensions of IDA* have been previously proposed: IDMOA* [3] and PIDMOA* [1].

IDMOA* focuses on a single objective at a time and computes scalar thresholds bounding the sucessive searches. Initially, when the first objective is considered, IDMOA* behaves like IDA*, computing only the values of the first component in the cost vector. Unlike IDA*, IDMOA* continues the search until all the solutions with the minimum value in the first component are located, so it is guaranteed that a Pareto-optimal solution has been found. Then, the rest of the objectives are considered sequentially in the same way, but including an upper limit for each one, given by the maximum value of the objective taken into account in the solutions found so far.

IDMOA* presents two main drawbacks. Since it focuses on a single objective at a time, information concerning the remaining objectives is discarded, which results in redundant re-expansions of nodes in the following iterations. Also it may temporarily add dominated solutions to the solution set, inducing extra tests to be performed each time a new goal node is located. On the other hand, since tests against the threshold just compare two scalar values, they can be computed very efficiently.

The other approach, PIDMOA*, takes into account all the objectives simultaneously, and each threshold is computed as the set of non-dominated cost vectors of the nodes discarded at the previous iteration. Search is discontinued when the cost vector of the node is dominated by any vector in the current threshold or is dominated by previously found solutions. Since PIDMOA* processes all the objectives at once, it drastically decreases node re-expansion. However, since the threshold is usually multi-vectorial, dominance tests are heavily time-consuming.

IPID —the proposal here presented— also maintains a vectorial threshold in order to decrease node re-expansion. However, in order to decrease the number of dominance tests performed by PIDMOA*, IPID reduces this multi-vectorial threshold to a single vector before search is actually performed in the next iteration. This value is the *ideal point* of the cost vectors of the nodes discontinued at the current iteration.

At the first iteration, IPID uses the heuristic vectors of the start node as the first threshold set. Then it performs a depth first search bounded by the *ideal point* of this threshold set. When the path currently being explored is dominated by any previously found solution, then this path is fully discarded, since it cannot lead to a Pareto-optimal solution. When the path is *strictly worse* than the threshold, then it is discarded and its cost vectors are used to compute the next threshold. In fact, requiring just a *dominance* test to discard a path may result in endless cycles when computing iteration thresholds. For example, let us consider a graph with heuristic function $\forall n\, \vec{h}(n) = \vec{0}$ and a start node s having two successor nodes s_1 and s_2 with $\vec{g}(s_1) = (1,2)$ and $\vec{g}(s_2) = (2,1)$. IPID will use $\vec{h}(s) = (0,0)$ as the initial threshold. At the first iteration, it will expand s, generating nodes s_1 and s_2. Since *threshold*$= (0,0) \prec (1,2) = \vec{g}(s_1)$ and *threshold* $= (0,0) \prec (2,1) = \vec{g}(s_2)$, search is discontinued at both nodes and a new ideal point is computed as *threshold* = *idealPoint*$\{(1,2)(2,1)\} = (1,1)$. The next depth-first search re-expands the start node s, generating g_1 and g_2. But now current threshold (1,1) dominates both $\vec{g}(s_1)$ and $\vec{g}(s_2)$; should IPID check just dominance, search would be discontinued again at both nodes, a threshold (1, 1) would be again computed, and the algorithm would be trapped in an endless computation. However, by applying the *strictly better* check, IPID ensures that a threshold is always different from the previous one. In our example, since (1,1) is not strictly better than (1,2) nor (2,1), both nodes s_1 and s_2 will be expanded in the second iteration.

If the current path is not discontinued, then IPID performs the *goal* test on the leaf node of the current branch. If it is a goal node, then its cost vector is added to the current solution set.

PIDMOA* only finds non-dominated solutions [1]. However, both IDMOA* and IPID may temporarily add dominated solutions to the solution set C^*, which will be discarded at later steps. This issue is present in the example in section 3.1

Taking into account these considerations, the complete pseudocode of IPID is as shown in table 1.

IPID (G, s, Γ)
SOL = $\varnothing$; ThresholdSet = *nodomset*($\{\vec{h}(s)\}$)
WHILE ThresholdSet $\neq \varnothing$
    $\vec{threshold}$ = *IdealPoint* (ThresholdSet)
    (ThresholdSet,SOL) = DFS (s, $\vec{threshold}$, SOL)
return (*nodomset*(SOL))

DFS (node,$\vec{currentTh}$, SOL)
ndomv = $\{\vec{f}(node) \in F(node) \mid (\nexists(\gamma, P^*(\gamma)) \in SOL \mid P^*(\gamma) \preceq \vec{f}(node))\}$
IF (ndomv = $\varnothing$) THEN return ($\varnothing$, SOL);
ndomv = $\{\vec{f}(node) \in F(node) \mid \neg(\vec{currentTh} \ll \vec{f}(node)))\}$
IF (ndomv = $\varnothing$) THEN return (F(node),SOL);
IF (node $\in \Gamma$) THEN
    SOL = SOL $\cup$ {(node, $\vec{f}$(node))}
    return ($\varnothing$, SOL)
ELSE
    ThresholdDFS = $\varnothing$
    successors = expand_node (node)
    FOR each n in successors DO
        (ThresholdRT,SOL) = DFS(n, $\vec{currentTh}$, SOL)
        ThresholdDFS = *nodomset* (ThresholdDFS $\cup$ ThresholdRT)
    return (ThresholdDFS, SOL)

Table 1. Algorithm IPID

Function IPID computes the threshold for each iteration by calculating the ideal point of cost vectors at nodes where search was discontinued. Then it performs the corresponding depth-first searches by calling the DFS function. This function returns a pair (*next-threshold*, SOL), being *next-threshold* the set of vectors used to compute the next single-vector threshold (its *ideal point*), and being SOL the set of solutions (cost vectors) found so far.

DFS function behaves as mentioned before: it discards nodes dominated by any previously found solution; it discards nodes strictly worse than the current threshold, considering their cost vectors for computation of the next threshold; it add cost vectors of the goal nodes to the SOL set; and expands the node if none of these conditions arise, by recursively calling the DFS function.

An example of IPID in action is presented in the next subsection.

3.1 Example

Figure 1 shows a simple bi-objective tree search problem, where each node is labelled with its heuristic vector and each arc is labelled with a single cost vector. The set of goal nodes Γ includes γ_1, γ_2 and γ_3. The paths to γ_2 and γ_3 are both Pareto-optimal solutions, while γ_1 is dominated by γ_2. Notice that although the whole search tree is depicted at each iteration, this tree will be expanded in a depth-first manner, from left to right.

The first iteration of IPID is depicted in Figure 2(a). The first threshold is $\vec{h}(s)$, that is, (0,0). Then a depth-first search bounded

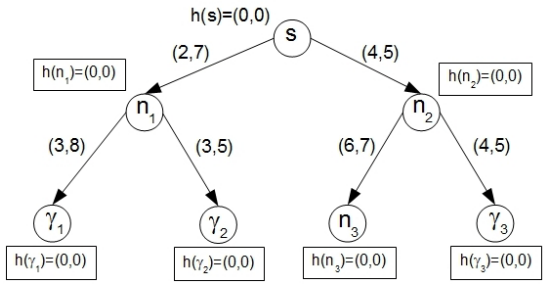


Figure 1. Multiobjective problem

by (0,0) is started. Since $\vec{f}(s)$ is not strictly worse than the current threshold, node s is expanded. Since the current threshold is strictly better than $\vec{g}(n_1)$ and $\vec{g}(n_2)$, search is discontinued at both nodes and a new threshold is computed by calculating the ideal point of $\vec{g}(n_1)$ and $\vec{g}(n_2)$.

The second iteration of IPID, shown in Figure 2(b), uses the vector (2,5) (strictly worse than the previous threshold (0,0)) as the new bound. The start node is expanded. Since the cost vectors of n_1 and n_2 are not strictly worse than the current threshold (even though it dominates them), both nodes are expanded. However, nodes at the leaf level are strictly worse than (2,5), so search is fully discontinued at this iteration. The next ideal point is computed as indicated, resulting in vector (5,10). A graphical representation of IPID and PIDMOA* thresholds is depicted in Figure 2(b). PIDMOA* would expand nodes outside the boundaries of the squares defined by each one of the cost vectors included in its threshold. On the other hand, IPID just defines a single vector as a threshold, simplifying *discontinuity* tests.

The third and last iteration of IPID is shown in Figure 2(c). The start node is expanded, as well as n_1 and n_2. Since $\vec{g}(\gamma_1)$ is not strictly worse than the current threshold (5,10), γ_1 is found to be a goal node and its cost vector (5,15) is added to SOL (even though it is a dominated solution). γ_2 is also generated and its cost vector (5,12) is added to the solution set, but since it dominates the previous solution, vector (5,15) is excluded from SOL. Node n_3 is generated, but its cost vector is dominated by a previously found solution (5,12), so it is fully discarded. Finally node γ_3 is also added to the solution set.

The next threshold is empty, so IPID finishes, returning the whole set of Pareto-optimal solutions.

3.2 Properties of IPID

We will make the following assumptions: (i) the graph G is connected and its branching factor is bounded; (ii) there exists at least one solution, i. e., a path from s to a node $\gamma \in \Gamma$; (iii) there exist positive numbers ε_i $(1 \leq i \leq q)$ such that for every i and for every edge cost $\vec{c}$ in G, $\vec{\varepsilon}_i \leq c_i$; (iv) all heuristic values $\vec{h}(n)$ are non-negative; (v) the heuristic function $H(n)$ is admissible. These assumptions are equivalent to those presented for IDMOA* and PIDMOA* in [1].

Lemma 1 *For every iteration i, $\vec{threshold}_i \ll \vec{threshold}_{i+1}$.*

Proof: Let us assume the contrary; then there exists a component j such that $\vec{threshold}_{i+1}(j) \leq \vec{threshold}_i(j)$. However, $\vec{threshold}_{i+1}$ is the ideal point of the next-threshold set T of costs of all nodes where search was discontinued at step i. By definition of IPID, for each $\vec{t} \in T$ and every component j we have $\vec{t}(j) > \vec{threshold}_i(j)$ and hence $\vec{threshold}_{i+1}(j) > \vec{threshold}_i(j)$, resulting in a contradiction. Therefore $\vec{threshold}_i \ll \vec{threshold}_{i+1}$.

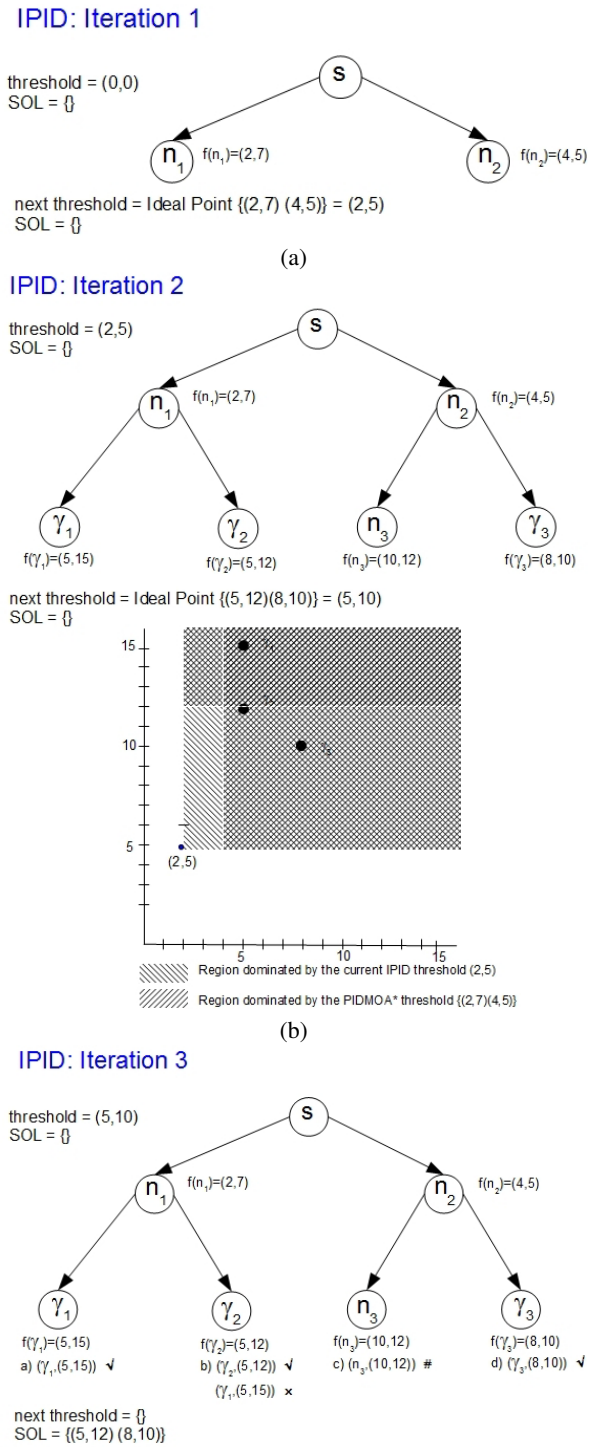


Figure 2. Example of IPID (a) 1^{st} iteration (b) 2^{nd} iteration (c) $3^{rd}iteration$

Lemma 2 *At anytime during the process of IPID, for every non-dominated solution path $P^* = (s, n_1, \ldots, n_i, \ldots, \gamma)$ and every subpath $P_i^* = (s, n_1, \ldots, n_i)$, there exists an $\vec{f}(P_i^*)$ such that $\vec{f}(P_i^*) \preceq \vec{f}(P^*)$.*

Proof: Trivial from the definition of admissibility in $H(n)$.

Lemma 3 *When IPID finishes, $C^* \subseteq SOL$, that is, every non dominated solution will be eventually found by IPID.*

Proof: For every non-dominated solution path P^*, there is at least a node belonging to this path which has been expanded by IPID at a given iteration. This is trivial for the first iteration, since the start node belongs to every path (particularly the non-dominated solution paths) in the search graph and $\exists \vec{h}(s) \in H(s) \mid \vec{threshold}_0 \preceq \vec{h}(s)$ and $\vec{h}(s)$ is not *strictly-worse* than $\vec{threshold}_0$ at this iteration. From lemma 1, we have that $\vec{threshold}_0 \ll \vec{threshold}_1 \ll \ldots \ll \vec{threshold}_n$, being n the number of iterations performed by IPID. That is, the start node will be expanded at each iteration.

Let us suppose that IPID finishes and the cost $\vec{c}^*$ corresponding to the non-dominated solution path P^* has not been discovered. Since IPID finishes, its final threshold is $\varnothing$. We know that a subpath $P_i^* \subseteq P^*$ has been expanded (containing at least the start node). Let m the last node from this subpath and let us assume that $\vec{c}^* \notin SOL$.

Search may have been discontinued at node m by the following reasons:

- $\exists \vec{c} \in SOL$ such that $\vec{c}$ dominates all $\vec{f}(m)$. But from lemma 2, we know that there exists a $\vec{f}(m)$ such that $\vec{f}(m) \preceq \vec{f}(P^*) = \vec{c}^*$ so $\vec{c} \prec \vec{c}^*$ and $\vec{c}^*$ would not be a nondominated solution cost, contrary to the assumption.
- Let $\vec{t}_{final}$ be the threshold for the last iteration of IPID. Search is discontinued because for all $\vec{f}(m)$, $\vec{t}_{final} \ll \vec{f}(m)$. In this case, IPID would add $\vec{f}(m)$ to the *next-threshold* set used to compute the following ideal point; so *next-threshold* is not empty and IPID does not terminate yet, so we have arrived again at a contradiction.

There are no more possibilities to discontinue search at node m, so the results holds.

Theorem 1 *IPID always finishes and at its termination $SOL = C^*$.*

Proof: Firstly, let us prove that IPID always finishes. Since by assumption the branching factor of the graph is bounded and each component of the cost vector is bounded from below by a possitive number ε_i, it is obvious that at every step i of deepening —given by a threshold $threshold_i$— the explored graph is finite and search finishes. On the other hand, by lemma 1, the sequence of thresholds is strictly increasing in every component j, and by assumption the increase is at least ε_j. Let $\vec{c}_{max} = (v_1, \ldots, v_k)$, where $v_i = \max\{y_j\}$ and y_j is the j-th component for every $\vec{f}(P_\gamma^*)$ included in C^*. Then for all $\vec{f}(P_\gamma^*) \in C^*$ it holds that $\vec{f}(P_\gamma^*) \prec \vec{c}_{max}$. Then each expanded path will reach a cost of $\vec{c}_{max}$ in at most $\lceil \frac{\max\{v_j\}}{\min\{\varepsilon_j\}} \rceil$ steps. At step $\lceil \frac{\max\{v_j\}}{\min\{\varepsilon_j\}} \rceil$, each expanded node n will verify that there exists $\vec{f}(P_\gamma^*) \in C^*$ such that $\vec{f}(P_\gamma^*) \prec \vec{f}(n)$ and, as a result of this, the threshold for the next iteration will be empty, so IPID will finish.

Now, by lemma 3, we know that at the termination step $C^* \subseteq SOL$. But there is an explicit final check to guarantee that vectors in SOL are nondominated; so dominated solutions that eventually could have been put into SOL will be discarded and the algorithm will return exactly the set C^*.

4 Empirical Evaluation

4.1 Setup

To perform the empirical evaluation of the three multiobjective iterative deepening approaches, an extensive set of random problems was generated. Each problem consists of an infinite binary tree, where each arc was labelled with a bidimensional cost vector. Values for each component of the vector varies in the integer range [1,50]. Both objective values are calculated using a uniform random distribution. A single null heuristic function ($\vec{h}(n) = \vec{0}, \forall n$) was used in every problem (in general, trends are not affected by the use of simple heuristics on random problems). According to the goal nodes, they were located at a fixed depth for each instance. Goal depths were set at levels 8, 10, 12, 14, 16, 18, 20 and 22. Tests also parameterized the number of goal nodes considered at each fixed depth. This number was reflected as a percentage of the nodes at the fixed depth.

So a single problem consists of an infinite binary tree, with bidimensional cost vectors, costs in the range [1,50] and correlation zero, and goal nodes located at a fixed depth (even values from 8 to 22). A percentage of the nodes at the fixed depth are goal nodes (percentages comprises 1%, 4%, 7%, 10%, 25%, 40%, 60% and 80%). The cost vectors of all these goal nodes are not necessarily Pareto-optimal solutions.

For each pair (*solution depth*, *percentage of goal nodes*), a group of five solution sets was generated. Each solution set was related to a different binary tree. Infinite random trees were generated using the efficient scheme described in [5], that better parametrizes solution depth and number of solutions..

4.2 Results

This section shows the results obtained by solving the problems described in subsection 4.1. The experiments were performed on a computer with two processors Six-Core AMD Opteron 2435 2600MHz and 64 GB of main memory. The algorithms were implemented with LispWorks Enterprise Edition 6.0, running on Windows Server 2008 R2 Enterprise 64bits . Tested algorithms were IDMOA*, PIDMOA* and IPID. Results for each triple (*algorithm*, *solution depth*, *percentage of goal nodes*) are averaged for 5 different problems with their corresponding solution sets.

Several figures regarding time requirements (in seconds) are depicted. Figures 3(a) and 3(b) show the results for problems with varying solution depth and 4% and 80% of goal nodes respectively, located at these solution depths. Figures 4(a) and 4(b) show the results for problems with fixed solution depth at levels 16 and 22 respectively, varying the percentage of goal nodes at these solution depths.

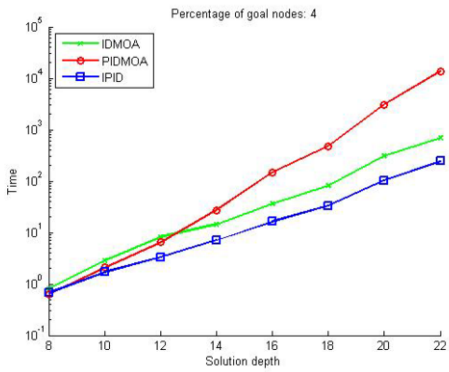
Additional figures are included analyzing several other performance measures related to the computation of two sample problems:

- Figure 5 shows the expanded nodes per iteration
- Figure 6 shows the size (number of vectors) of both C^* and threshold sets at each iteration. Related to IDMOA*, the threshold size is symbolic (value 1), since the threshold for this algorithm is scalar. Threshold size for IPID is constant (1 vector) since the *ideal point* computed at each iteration is a single vector.

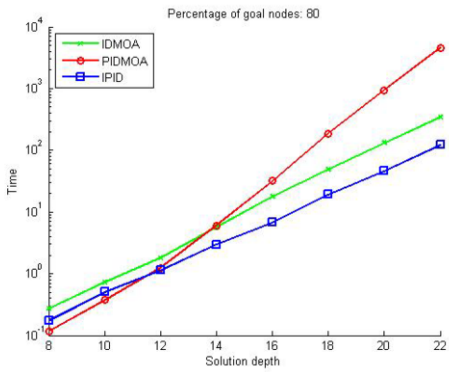
All the figures regarding time requirements use logarithmic scale for the vertical axis. The remaining figures maintain a linear scale for the vertical axis.

4.3 Discussion

All the algorithms tested in this paper share a common iterative deepening nature, having linear space requirements, so our evaluation of the experiments' results will focus mainly on time performance. Previous works on performance of single objective problems, like [10],

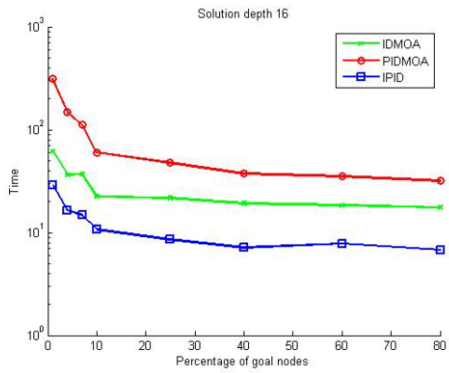


(a)

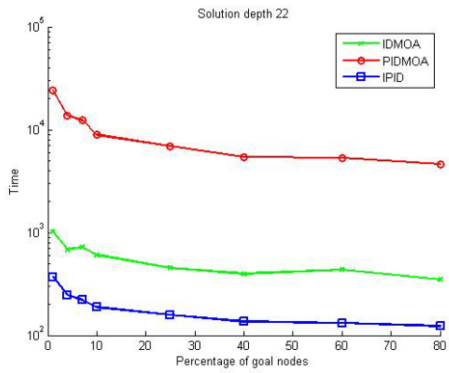


(b)

Figure 3. Time requirements (in seconds) with logarithmic scale for IDMOA*, PIDMOA* and IPID with (a) 4% of goal nodes (b) 80% of goal nodes



(a)



(b)

Figure 4. Time requirements (in seconds) with logarithmic scale for IDMOA*, PIDMOA* and IPID with (a) Solution depth 16 (b) Solution depth 22

consider the number of expanded nodes as a good proxy for execution time, since it is assumed constant processing time per node. It is a reasonable assumption, because just a scalar comparison must be added to node expasion operations. However, this does not apply to many multiobjective algorithms, where the cost vector of a node must be compared against both C^* and threshold sets. Since these sets have a variable size, the processing time per node may vary considerably and execution time does not depend just on the number of expanded nodes.

Figures 5(a) and 5(b) show the number of nodes per iteration expanded by the algorithms in two sample problems. Since IDMOA* processes the objectives one at a time, discarding most of the information related to the remaining objectives, it increases the number of iterations (figure 5), and so does the number of re-expansion of nodes compared to its counterparts PIDMOA* and IPID. This applies mainly at lower rates of goal nodes in the fixed solution depth. However, since the number of expanded nodes is greater in IDMOA*, this does not lead to greater time requirements, as it can be shown in Figures 3 and 4. In fact IDMOA* performs better than PIDMOA* in all but the shallower searches.

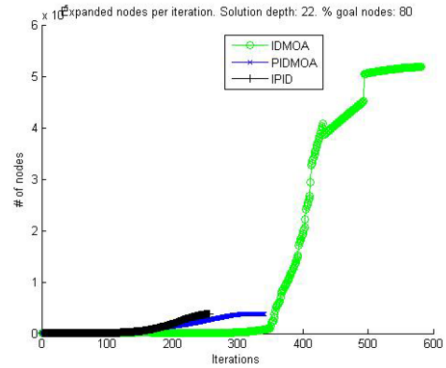
The reason for this behaviour relies on the simplicity of the scalar tests performed to test the *discontinuity* condition. PIDMOA* maintains a threshold usually containing several cost vectors. The evolution of the size of the threshold sets for two sample problems is depicted in Figure 6. Each expanded node has to be compared against the C^* set as well as the threshold set, in case it is not dominated by any located solution. This results in a heavy time overload, though the number of nodes expanded by PIDMOA* is commonly smaller than the ones expanded by IDMOA*.

IPID takes the best ideas from both approaches. It reduces considerably the number of iterations (compared to IDMOA*) by maintaining not an scalar threshold, but a vectorial one. This results in faster advances of the threshold and a smaller amount of re-expansions of nodes. But since vectorial dominance tests may drastically decrease the performance of the algorithm, IPID keeps the size of the threshold set to a minimum (Figure 6). While the threshold set of IPID remains constant, the threshold for PIDMOA* increases considerably as the algorithm deepens in the search tree. This threshold set decreases (as well as dominance tests) as Pareto-optimal solutions are located, which results in prunes of the search space.

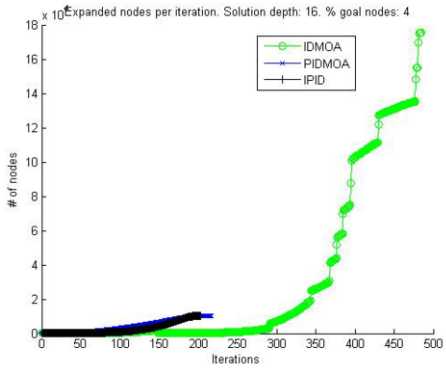
Figures shown here related to expanded nodes and size of threshold and C^* sets involve just two sample problems. However, the same trends can be detected for different problem instances with varying solution depths and percentages of goal nodes.

Figures 3 and 4 show the time requirements for several problem sets. Figure 3(a) analyzes the algorithms when solving problems with a 4% of goal nodes at a fixed depth, varying from 8 to 22. PIDMOA* is found to behave more efficiently at lower depths, but as the depth of the solutions increases, IPID becomes the faster algorithm. The same trend is observed in Figure 4(b). When compared to IDMOA* in terms of time requirements, IPID proved to be up to three times faster than IDMOA*.

Figures 4(a) and (b) show the time requirements related to problems with solution depth 16 and 22, varying the percentage of goal nodes. IPID remains the more efficient algorithm, increasing its difference with IDMOA* and PIDMOA* at higher solution depths. As the percentage of goal nodes increases, so does the efficiency of

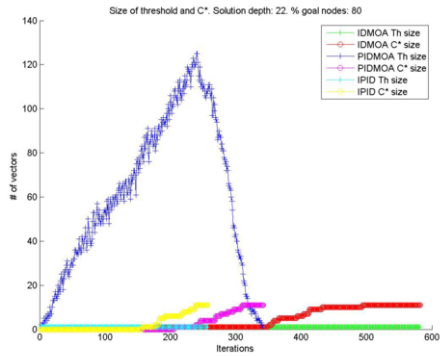


(a)

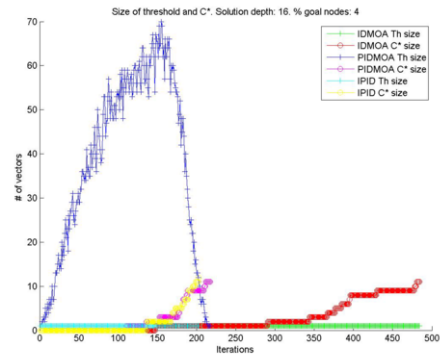


(b)

Figure 5. Expanded nodes per iteration for problems with (a) 80% of nodes at depth 22 being goal nodes (b) 4% of nodes at depth 16 being goal nodes



(a)



(b)

Figure 6. Size of threshold and C^* sets per iteration for problems with (a) 80% of nodes at depth 22 being goal nodes (b) 4% of nodes at depth 16 being goal nodes

the algorithms. This is due to the density of Pareto-optimal solutions. The more goal nodes we have, the more likely is to find non-dominated paths, which results in larger prunes of the space search. In our testbed, the Pareto-optimal solution set becomes saturated approximately with 40% of goal nodes. At larger percentages, no significative improvements are achieved by any algorithm.

5 Conclusions and Future Work

This paper presents IPID, a new extension of the iterative deepening paradigm to the multiobjective case. The algorithm is proven to be admissible, i.e., to terminate and return all the Pareto-optimal solutions. IPID aims to improve previous proposals by considering all the objectives at once, but minimizing the number of vectorial comparisons performed. This is achieved by keeping a single-vector threshold (the *ideal point*) to control the sequence of deepenings.

A detailed testbed over infinite random binary trees with biobjective cost vectors, varying the solution depth and the number of goal nodes, shows that IPID outperforms both IDMOA* and PIDMOA* in terms of time requirements. Results also raise some questions like the deep impact of dominance tests on performance, or the advantages of having a great number of goal nodes to prune wider areas of search space.

Future work includes an extension of this comparison to other depth-first multiobjective algorithms outside the iterative deepening family, like Branch and Bound.

ACKNOWLEDGEMENTS

This work has been partially funded by Consejería de Innovación, Ciencia y Empresa. Junta de Andalucía (España) - P07-TIC-03018 and TIN2009-14179, Plan Nacional de I+D+i, Gobierno de España.

REFERENCES

[1] J. Coego, Lawrence Mandow, and J. L. Pérez de la Cruz, 'A new approach to iterative deepening multiobjective A*', in *AI*IA 2009, LNCS 5883*, pp. 264–273, (2009).

[2] Pallab Dasgupta, P.P. Chakrabarti, and S.C. DeSarkar, *Multiobjective Heuristic Search*, Vieweg, Braunschweig/Wiesbaden, 1999.

[3] S. Harikumar and Shashi Kumar, 'Iterative deepening multiobjective A*', *Information Processing Letters*, **58**, 11–15, (1996).

[4] Richard E. Korf, 'Iterative-deepening A*: an optimal admissible tree search', in *Proc. of the IX Int. Joint Conf. on Artificial Intelligence (IJCAI'85)*, pp. 1034–1036, (1985).

[5] Richard E. Korf and David Maxwell Chickering, 'Best-first minimax search', *Artif. Intell.*, **84**(1-2), 299–337, (1996).

[6] E. Machuca and Lawrence Mandow, 'Multiobjective route planning with precalculated heuristics', in *Proc. of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)*, pp. 98–107, (2011).

[7] Ioannis Refanidis and Ioannis Vlahavas, 'Multiobjective heuristic state-space planning', *Artificial Intelligence*, **145**, 1–32, (2003).

[8] E. Rollon and J. Larrosa, 'Constraint optimization techniques for multiobjective branch and bound search', in *Lecture Notes in Economics and Mathematical Systems, Vol. 618*, pp. 89–98, (2009).

[9] Francis Sourd and Olivier Spanjaard, 'A multiobjective branch-and-bound framework: Application to the bi-objective spanning tree problem', *INFORMS Journal on Computing*, **20**(3), 472–484, (2008).

[10] W. Zhang, *State-Space Search: Algorithms, Complexity, Extensions, and Applications*, Springer, 1999.