Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2017-013 September 1, 2017

The Interval Programming Model Solution Algorithm Experimentation Tools and Results

Michael R. Benjamin

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

The Interval Programming Model Solution Algorithm Experimentation Tools and Results

Michael R. Benjamin
Computer Science and Artificial Intelligence Laboratory
Department of Mechanical Engineering
Cambridge MA 02139
Email: mikerb@csail.mit.edu

August 30, 2017

Abstract

Interval programming (IvP) is model for representing multi-objective optimization problems along with a set of solution algorithms. This paper describes a set of IvP solution experiments run over randomly generated problem instances, using five different versions of the Recursive Interval Programming ALgorithm (RIPAL). The final version is the algorithm used most extensively in practice, with the first four provided mostly for comparison as the final version is built up in complexity. The full details of the algorithms are outside the scope of this paper, with the focus here being the experimental results, and the software tools and technique used in generating the problem instances. Additional tools are described for facilitating the experiments, including visualization tools, and tools for generating the plots and tables shown in this document. All software tools are available under an open source license, and all problem instances reported here are also available online. This document is meant to supplement other discussions on the IvP model, algorithm, and IvP applications to provide the detail of reporting that would not be possible due to length restrictions of other papers.

Contents

1 Introduction **4**
- 1.1 IvP Domains . . . 4
- 1.2 IvP Functions . . . 4
- 1.3 The IvP Solution Algorithm . . . 5
- 1.4 Ring Functions . . . 6
- 1.5 Ring Function Parameters for Reported Experiments . . . 8
- 1.6 Example Ring Functions . . . 8

2 IvP Experimentation Software Utilities **10**
- 2.1 The ippgen Utility for Generating IvP Problems . . . 10
- 2.2 The ippsolve Utility for Solving IvP Problems . . . 14
- 2.3 The ippview Utility for Rendering IvP Problems . . . 17
- 2.4 The isrplot Utility for Generating Plots from Results . . . 18
- 2.5 The isrtex Utility for Generating LaTex Tables from Results . . . 19

3 Trial 1: RIPAL Version 1 vs. Version 2 **21**
- 3.1 Trial 1A: RIPAL Version 1 vs. Version 2 (3 IvP Functions) . . . 21
- 3.2 Trial 1B: RIPAL Version 1 vs. Version 2 (4 IvP Functions) . . . 24

4 Trial 2: RIPAL Version 2 vs. Version 3 **27**
- 4.1 Trial 2A: RIPAL Version 2 vs. Version 3 (2 IvP Functions) . . . 27
- 4.2 Trial 2B: RIPAL Version 2 vs. Version 3 (3 IvP Functions) . . . 30

5 Trial 3: RIPAL Version 3 vs. Version 4 (10 IvP Functions) **33**
- 5.1 Trial 3A: RIPAL Version 3 vs. Version 4 (2 Dimensions) . . . 33
- 5.2 Trial 3B: RIPAL Version 3 vs. Version 4 (4 Dimensions) . . . 36

6 Trial 4: RIPAL Version 3 vs. Version 4 (20 IvP Functions) **39**
- 6.1 Trial 4A: RIPAL Version 3 vs. Version 4 (2 Dimensions) . . . 39
- 6.2 Trial 4B: RIPAL Version 3 vs. Version 4 (4 Dimensions) . . . 42

7 Trial 5: RIPAL Version 3 vs. Version 4 (Varying IvP Functions) **45**
- 7.1 Trial 5A: RIPAL Version 3 vs. Version 4 (2 Dimensions) . . . 45
- 7.2 Trial 5B: RIPAL Version 3 vs. Version 4 (4 Dimensions) . . . 48

8 Trial 6: RIPAL Version 4 vs. Version 5 **51**
- 8.1 Trial 6A: RIPAL Version 4 vs. Version 5 (3 Dimensions) . . . 51
- 8.2 Trial 6B: RIPAL Version 4 vs. Version 5 (4 Dimensions) . . . 54

9 Trial 7: RIPAL Version 4 vs. Version 5 (Varying IvP Functions) **57**
- 9.1 Trial 7A: RIPAL Version 4 vs. Version 5 (2 Dimensions) . . . 57
- 9.2 Trial 7B: RIPAL Version 4 vs. Version 5 (4 Dimensions) . . . 60

9.3 Data From Trial 7B - RIPAL Version 5 62

10 Trial 8: RIPAL Version 3 vs. Version 5 (Varying IvP Functions) 63
10.1 Trial 8A: RIPAL Version 3 vs. Version 5 (2 Dimensions) 63
10.2 Trial 8B: RIPAL Version 3 vs. Version 5 (4 Dimensions) 66

1 Introduction

Here we briefly review the key interval programming (IvP) model components and algorithm constructs to provide context for the experimental results reported in later sections. A full discussion of these topics as well as their motivation and application to autonomous decision making, is outside the scope of this paper. In each experiment reported in this paper, the experiment will be defined by how each of the five following aspects of the problem are set:

- The IvP domain (the number of dimensions and decisions in each dimension)
- The number of IvP objective functions (or simply IvP functions)
- The number of pieces in each IvP function
- The piece distribution in each function (uniform or otherwise)
- The algorithm type and algorithm parameters

Here we briefly review the topics relevant to the above parameters.

1.1 IvP Domains

The IvP model assumes a finite, uniformly discrete domain for each of its decision variables. A domain is defined by:

$$\mathbb{D} = (x_1, \ldots, x_n)$$

Each domain variable, x_i is given by $\{x_i : \min x_i, \max x_i, |x_i|\}$. For many of the experiments described in this paper, the domain has two variables, x and y, with 1000 elements for each variable, totaling one million distinct decisions. This domain would be described with string:

```
"x,0,999,1000:y,0,999,1000"
```

Similarly a 3D domain would be described with:

```
"x,0,999,1000:y,0,999,1000:z,0,999,1000"
```

Throughout this report, a domain of N dimensions is assumed to be similarly defined, with 1000 elements per variable.

1.2 IvP Functions

IvP functions are piecewise linear approximations of an underlying utility function. For each function, each element of the decision space is contained in exactly one piece of the function. A function may be uniform in terms of the size of each piece. Two examples of uniform functions are shown below with different sizes chosen for the uniform piece. A fundamental tradeoff exists between a higher number of pieces (greater accuracy but slower creation and solution time) and a lower number of pieces (less accuracy but faster creation and solution time).

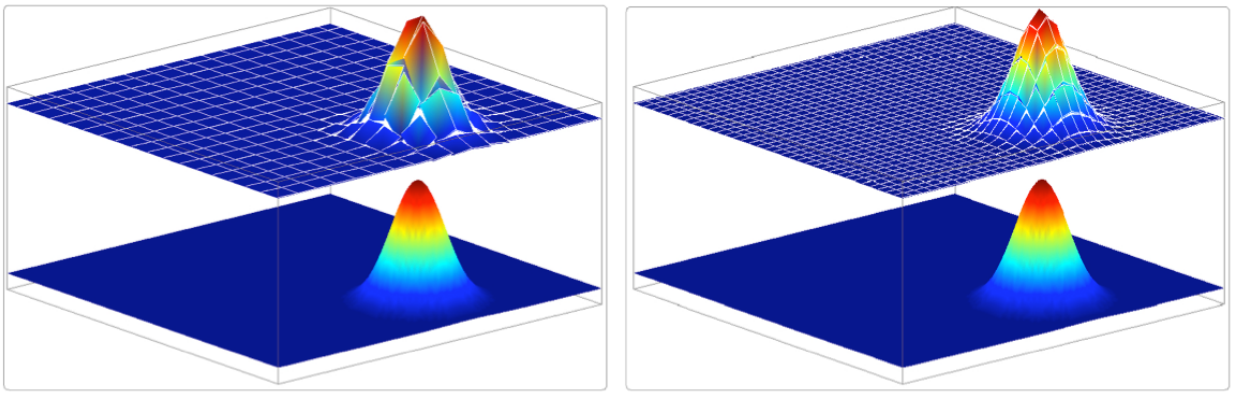

Figure 1: Two piecewise linear approximations of the same underlying objective function. The piece distributions here are uniform, but they need not be.

One way to mitigate the accuracy/speed tradeoff is to allow more pieces to be used in areas of the function that are highly nonlinear. The IvP function is typically not uniform. An example is shown below.

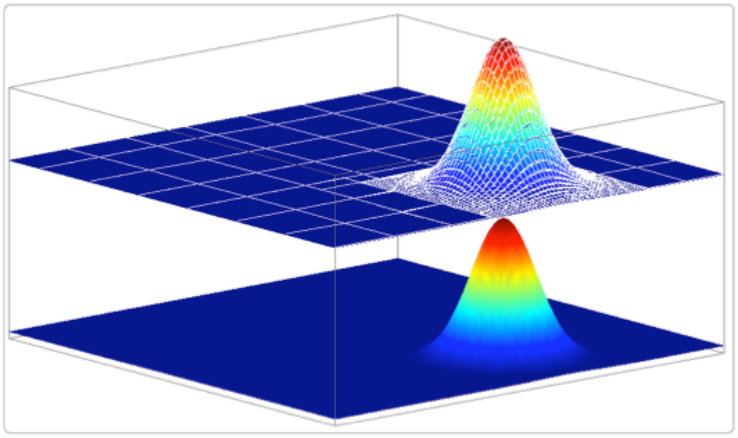

Figure 2: Non-uniform functions allow more pieces to represent areas of the function where the value is highly nonlinear.

The algorithm and process for creation of these functions is outside the scope of this report.

1.3 The IvP Solution Algorithm

Five IvP solution algorithms are reported in this paper. They are progressively more sophisticated and fast. They all treat the solution space the same - the set of possible combinations of a single piece from each objective function. The search structure is branch-and-bound, and the number of leaf nodes explored during the course of the solution process is included in the table of results for each experiment.

In each algorithm variation, the problem is comprised of k objective functions, $f_i(\mathbf{x})$, defined over a common decision space $\mathbf{x}$. Each objective function has a weight w. Once the weights and functions are defined, the problem reduces to a single objective function, where the solution of the problem is the element of the decision space that optimizes this function. Or more formally:

$$\mathbf{x}^* \quad = \quad argmax_{\mathbf{x}} \sum_{i=1}^{k} (w_i \cdot f_i(\mathbf{x})) \tag{1}$$

If each objective function has m pieces, the jth of m piece, from the ith of k objective functions is referred to as $p_{i,j}$. The IvP search algorithm proceeds by searching through the space of possible piece combinations, one from each of k piecewise linear functions. Each candidate solution is a vector $v = [v_1, \ldots, v_k]$ where each v_i is a piece index from the ith objective function, and $p_{(i,v_i)}$ represent the v_ith piece from the ith objective function. A tree structure is shown in Figure 3.

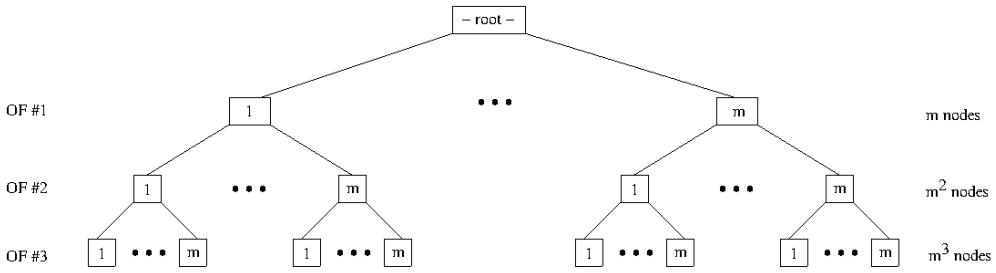

Figure 3: The search tree for $k = 3$ objective functions with m pieces each.

Each leaf node corresponds to candidate solution, one piece from each objective function. The five algorithms reported on this paper are variations of the same algorithm, but differ in whether and how they prune nodes of the tree during search. All are guaranteed to be globally optimal.

1.4 Ring Functions

In the experimental results reported here, we need the ability to generate reasonably challenging random functions for testing. Some desirable features are that they be non-convex and multi-modal. Occasionally it may be good if the optimal decision is a plateau of optimal decisions. Furthermore it would be good if the functions were easily defined by just a few parameters with ranges from which random values could be chosen to give suitably challenging set of test cases. There's no single right answer, but we chose a class of functions we call *ring functions*.

A *ring function* is built by placing a $n-$dimensional ring somewhere in $n-$space, by choosing both its center and radius, and then associating values to all points in the domain based on the distance to the ring. Simple examples are depicted below in Figure 4. More than one ring can be used as shown in Figure 5.

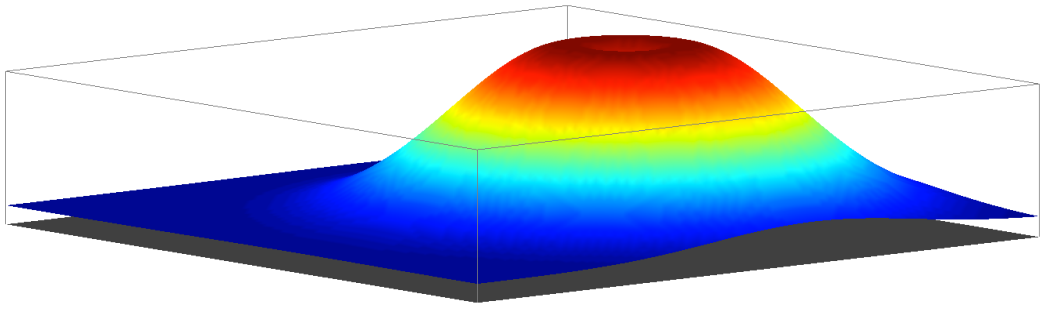

Figure 4: Example ring functions.

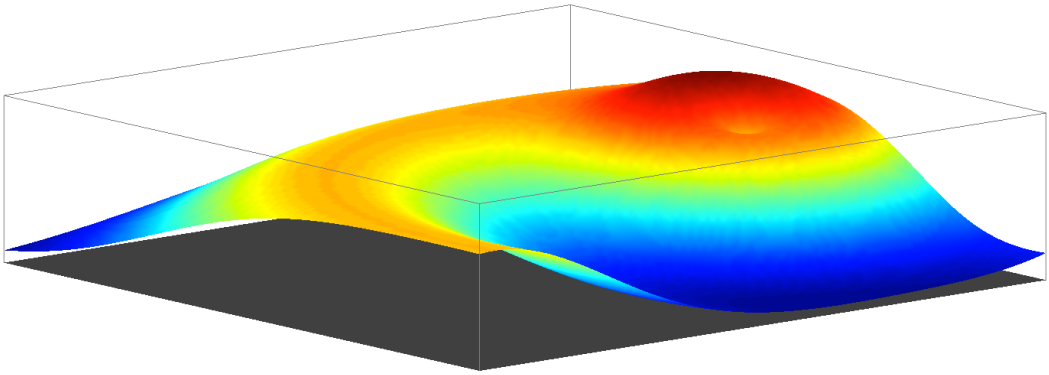

Figure 5: Example ring functions.

The following definition identifies the parameters of interest for generating sets of ring functions (over 2 variables by example). A *ring function* is a function with the following form:

$$f(x,y) = f_1(x,y) + \ldots + f_p(x,y) \tag{2}$$

Each of the p functions $f_i(x,y)$ correspond to a particular ring given by:

$$f_i(x,y) = ((1 - \frac{|\sqrt{(x-h)^2 + (y-k)^2} - \texttt{rad}|}{\texttt{max-dist}})^{\texttt{exp}} * \texttt{range}) + \texttt{base} \tag{3}$$

Each function $f_i(x, y)$ is equal to $g(r(x, y))$ given by:

$$r(x, y) = |\sqrt{(x - h)^2 + (y - k)^2} - \texttt{rad}| \tag{4}$$

and

$$g(x) = ((1 - \frac{r(x, y)}{\texttt{max-dist}})^{\texttt{exp}} * \texttt{range}) + \texttt{base} \tag{5}$$

The function $r(x, y)$ indicates a circle (ring) with radius `rad`, and returns the shortest distance of a point, (x, y) to the ring. The function $g(x)$ takes this distance and produces the desired value based on the following intuition. The center of each ring is set to be somewhere in the universe given by the Cartesian product of each variable's domain. The value of `max-dist` is the maximum distance in this universe, i.e., the length from corner to opposite corner of the universe. The value of $\frac{r(x,y)}{\texttt{max-dist}}$ is therefore always in the range $[0, 1]$. Subtracting this value from 1 and raising it to the exponent `exp` still leaves us with a value in the range $[0, 1]$. Multiplying this by `range` and adding it to `base`, ensures that each function, $f_i(x, y)$, is guaranteed to range over $[\texttt{base},\ \texttt{base} + \texttt{range}]$, and thus $f(x, y)$ has the range $[p(\texttt{base}), p(\texttt{base} + \texttt{range})]$. The actual range of the function may be quite smaller and is unknown based solely on the parameters.

1.5 Ring Function Parameters for Reported Experiments

In the experiments reported in this paper, all randomly generated functions are constructed in the following way:

- the number of rings is fixed at $p = 2$
- the center of each (h, k) is a uniformly random choice within the IvP domain
- the radius of each ring is a uniformly random value in the range $[5, 500]$
- the exponent of each ring is a uniformly random value in the range $[2, 25]$
- all functions are normalized to be in the range of $[0, 100]$

1.6 Example Ring Functions

Some example functions are shown in the below Figures.

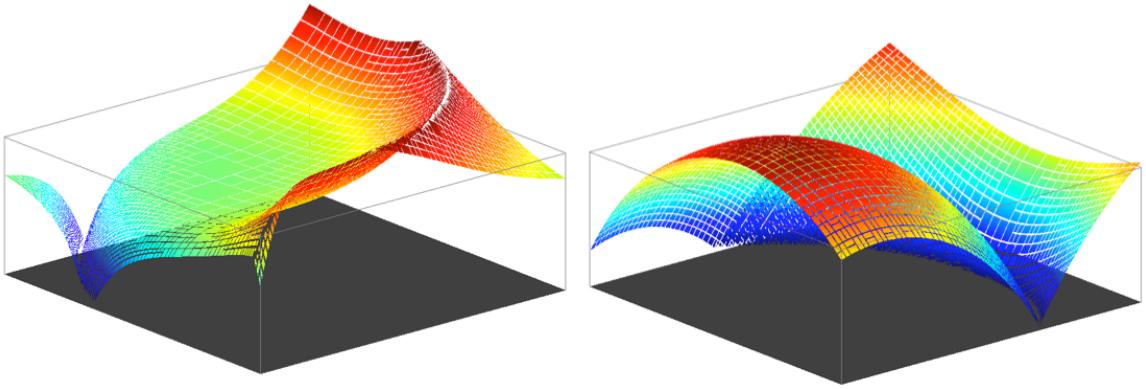

Figure 6: Example randomly generated ring functions

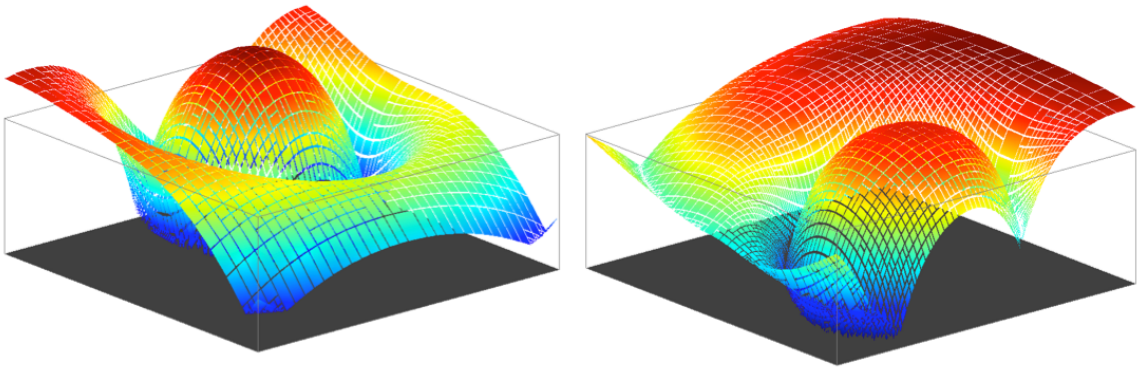

Figure 7: Example randomly generated ring functions

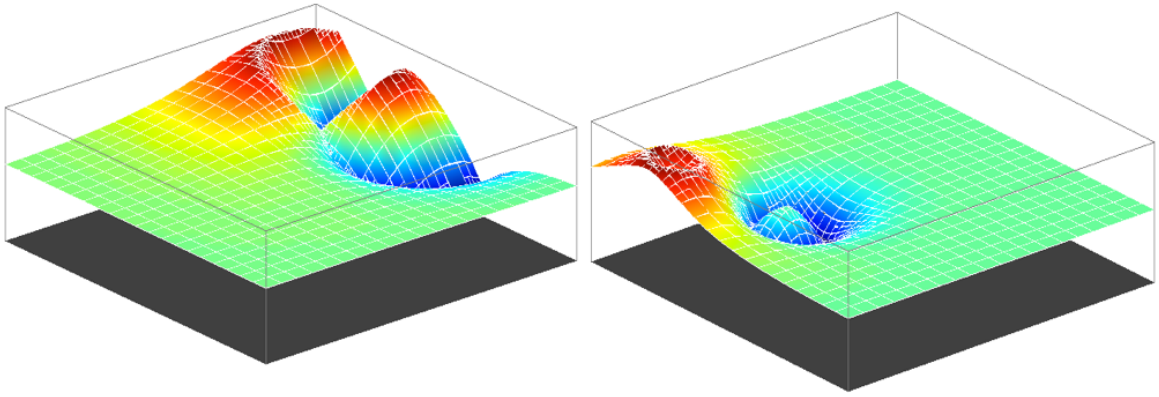

Figure 8: Example randomly generated ring functions

2 IvP Experimentation Software Utilities

This section describes five command line software utilities used for generating, solving, analyzing and rendering the IvP problems and results reported in this document. They include:

- `ippgen`: A utility for generating random instances of IvP problems, and storing the instance to a file (`.ipp` file).
- `ippsolve`: A utility for solving IvP problem using one of several available solution algorithms and storing the solution time and results to a result file (`.isr` file).
- `ippview`: A utility for rendering IvP problems in 2 dimensions.
- `isrplot`: A utility for reading a results (`.isr`) file and generating file suitable for generating plots in GNUPlot, a (`.dat`) file.
- `isrtex`: A utility for reading a results (`.isr`) file and generating a LaTex (`.tex`) file with a table of the results read from the results file.

For users interested in using these utilities to replicate the results of this document, please visit the below URL or contact the author:

`www.moos-ivp.org/benchmarking`

This site contains access to not only the tools described here, but also the shell scripts invoking these tools used to generate the problem instances and results reported here. The problem instances are randomly created, so even with the same tools and same scripts, some variance in the results is expected. If interested in the actual problem instance files reported in this document, they have been archived, and are downloadable from the same URL as above.

2.1 The ippgen Utility for Generating IvP Problems

The `ippgen` utility may be used for generating random IvP problems and storing them to a file for later solution testing. The user may specify (a) the number of dimensions (decision variables), (b) the number of IvP functions, (c) the number of pieces and piece distribution, and (d) parameters of the underlying objective function to be approximated.

On the command line a simple IvP problem can be created with the following command, essentially accepting the default values for all command line arguments:

```
$ ippgen
```

The above will generate a file, `file.ipp`, containing 1 objective function in 2 dimensions, with 100 uniform pieces, approximating a ring function with two randomly generated ring components.

2.1.1 Specifying the Decision Space

By default ippgen uses a decision space with two variables, x and y, each with 500 discrete decisions. The string representation of this decision space is:

```
"x:0:499:500#y:0:499:500"
```

For the variable x, the lowest possible value is 0. The highest value is 499, with 500 choices, creating a discrete domain of $[0, 1, 2, \ldots, 498, 499]$.

A different domain with four variables, w, x, y, z, each having 1000 choices could instead be specified on the command line with the `--dom` argument:

```
$ ippgen --dom=w:0:999:1000#x:0:999:1000#y:0:999:1000#z:0:999:1000
```

2.1.2 Specifying the Number of IvP Functions in the IvP Problem

By default ippgen will create an IvP problem with a single objective function. Higher numbers may be specified with the `--ipfs` argument:

```
$ ippgen --ipfs=20
```

For each generated IvP function, a different underlying objective function, e.g., ring function, is randomly generated.

2.1.3 Specifying the Number of Uniform Pieces in an IvP Function

All IvP functions are piecewise linear functions, but they need not be uniformly sized pieces. This is a key strength of the model which allows the function approximation to use more pieces in parts of the domain where the function being approximated is more nonlinear (see Figure 2). However, as a simple default, ippgen will create each IvP function in an IvP problem using a uniform piece, where each piece contains 50 elements of the decision space. This parameter is referred to as the *build_info*. The string representation of the default *build_info* is:

```
"uniform_piece=discrete@x:50,y:50"
```

This specifies two decision variables, x and y, with a uniform piece size of 50. If the domain, specified separately, doesn't also have these two variables, and have at least 50 elements, then ippgen will not generate an IvP problem.

If the default domain is used (Section 2.1.1), and the default *build_info* is used as above, then exactly 100 uniformly sized pieces will be generated. A different build configuration may be specified using the `--build` argument:

```
$ ippgen --build="uniform_piece=discrete@x:100,y:20"
```

The above would create 125 uniform pieces each of length 100 in the x dimension and 20 in the y dimension.

2.1.4 Specifying the Number of Smart Pieces in an IvP Function

IvP problems comprised of IvP functions with purely uniform pieces would not be very interesting to solve, especially if each function used the same sized uniform piece. In such problems, each piece would intersect exactly one piece from each of the other IvP functions and the search "tree" would only have a branching factor of one. Such functions would also likely be more wasteful and not as accurate an approximation of their underlying functions as they could be. In highly linear areas of the underlying functions, more pieces would be used than may be necessary. In highly nonlinear areas of the underlying function perhaps not enough pieces are used for ensuring a sufficiently accurate approximation (See Figure 1).

Non-uniform application of pieces is particularly effective when the builder of the function has insight into exactly where the nonlinear areas of the underlying function reside. As in the example shown in Figure 2, the nonlinear region is quite apparent, and therefore more pieces are dedicated to this region. The uniform pieces outside this could be made even larger without loss in precision of the approximation. However, the allocation of resource shown in the figure is clearly much more efficient then if the same number of total pieces were used in a purely uniform manner.

The `ippgen` tool creates *randomly* generated functions. Since the underlying functions are generated randomly, it can be tricky to infer just where the nonlinear regions reside and where to allocate more pieces. So instead, the `ippgen` utility uses an algorithm referred to as *smart refinement*. The algorithm begins as if the IvP function were to consist solely of uniform sized pieces. However, as each piece is fitted with a linear interior function, points inside the piece are sampled to gauge how well the linear function fits the underlying function. Those with poor fits are given a higher priority score and inserted into a priority queue. After the initial uniform function is built, the priority queue is repeatedly popped with the highest priority piece - the one with the poorest known fit. This piece is split, with both pieces re-fit with a linear interior function and both pieces re-analyzed for their fit. Both pieces are then re-inserted to the priority tree. The algorithm continues this way until the number of allocated "smart" pieces has been exhausted. The number of smart pieces may be specified in the `build_info` command line argument:

```
$ ippgen --build="queue_levels=8#smart_amount=400"
```

This invocation of `ippgen` would build 500 total pieces. The first 100 pieces are built of uniform size as described earlier. The remaining 400 pieces are allocated to the smart refinement algorithm. The size of the priority queue is indicated by the number of levels in the priority queue, in this case 8 levels or $2^8 = 256$ nodes.

2.1.5 Specifying the Grid Size in an IvP Function

An IvP function has an associated grid structure to facilitate certain queries key to the IvP solution algorithm. One query is the intersection query - for a given piece, presumably from another IvP function, which pieces of this IvP function intersect that piece? With the grid structure, the intersection test proceeds first by finding which grid cell(s) intersect with the given piece, and then perform an intersection test on only those pieces in the grid cell(s). This structure is vital to the performance of the solution algorithm. Choosing this grid structure properly is also very important. There is no single formula for choosing the proper grid parameters for any given IvP function a priori. However, in cases where an IvP function is created with an initial uniform piece distribution, followed by further refinement, a useful heuristic in practice is to align the grid structure with the initial uniform pieces.

The default grid structure employed by `ippgen` uses grid cell extents twice the size of the default uniform piece size. The string representation of this is:

```
"uniform_grid = discrete @ x:100,y:100"
```

There are two general rules of thumb. The first is to choose a grid structure that each piece resides in exactly one grid cell. The IvP solution algorithm does not depend on this, but will indeed provide faster solutions all things being equal if this condition holds. The second goal is to aim for a "reasonable" ratio of pieces contained in each grid cell. A ratio of one piece per grid cell will not allow the grid structure to provide any benefit over simply performing queries over the pieces directly. And a ratio of too many pieces per grid cell also reduces the grid structure effectiveness. In practice this ratio can be experimented offline to quickly find an effective setting.

In IvP functions created by initial uniform piece distribution, followed by further refinement such as the smart refinement technique, a common heuristic is to match the grid size to the size of the initial uniform pieces. For example, in the following invocation of `ippgen`, the grid structure is aligned with the initial uniform pieces:

```
$ ippgen --dom=x:0:499:500#y:0:499:500          \
  --build="uniform_piece=discrete@x:50,y:50#    \
  queue_levels=15#uniform_grid=discrete@x:50,y:50#smart_amount=400"
```

2.1.6 Generating a Batch of IvP Problems

By default `ippgen` will generate a single IvP problem, and by default in a file named `file.ipp`. Typically this tool is used to generate a large batch of similarly configured IvP problems. With randomization the underlying objective function, approximated by the IvP function, will vary. And if smart refinement is used, the piece distribution beyond the initial uniform pieces will also also vary between IvP functions and problems. Use the `--amt` argument on the command line to specify the desired number of files. For example:

```
$ ippgen testfile.ipp --amt=100
```

This will generate 100 files named:

```
testfile_00.ipp
...
testfile_99.ipp
```

2.1.7 Specifying the Underlying Objective Function

A key part of the ippgen utility, in addition to specifying the parameters of the IvP function, is the ability to specify the parameters of the underlying objective function to be approximated. The ippgen utility primarily uses the ring functions described earlier in Section 1.4. This allows for a wide variety of functions with just a few variations of input parameters.

The parameters used for generating IvP problems reported in this paper were described in Section 1.5. The string representation of these set of parameters is:

```
"rings=2 # radlow=5 # radhigh=500 # explow=2 # exphigh=25 # gradient=linear"
```

The center of each ring component is located at a uniformly random location in the IvP domain, and all functions are normalized to the range of $[0, 100]$. Otherwise the above parameters may be specified on the command line to ippgen. For example:

```
$ ippgen -aof="rings=4 # radlow=2 # radhigh=20"
```

2.1.8 Seeding the Random Number Generator

By default the random number generator in ippgen is not seeded. To seed the random number generator, use the `--seed`, or simpley the `-s` parameter alongside any other parameters.

```
$ ippgen --seed
```

2.2 The ippsolve Utility for Solving IvP Problems

The ippsolve utility may be used for reading in a previously created IvP problem from a file, solving the problem, and saving the solution and solve-time results to a file. On the command line, the user may also specify (a) the solve algorithm version, (b) a solution threshold, (c) an initial solution algorithm, (d) the number of times to solve the problem (for finer grained timing measurements on simpler problems), and (e) a file to which the results will be added.

On the command line, assuming `file.ipp` already exists, holding an IvP problem instance, `ippsolve` may be run as follows (accepting all command line defaults):

```
$ ippsolve file.ipp
file.ipp: Building...  Done. Solving... Done.
```

This output is not very informative, but if run with the `--verbose` option or if an output file is specified, full solve results may be seen.

2.2.1 Saving ippsolve Solution Results to a File

The `ippsolve` utility may produce solution results to a file, if a file with the suffix `.isr` is specified. The `.isr` suffix is derived from "interval-programming solution results". For example:

```
$ ippsolve file.ipp file.isr
file.ipp: Building...  Done. Solving... Done.
$ cat file.isr
ipp=file.ipp, cpu_time=0.34, dim=2, isol=false, ipfs=2, alg=0, pcs=100, \
              leafs=6, thresh=100, decision=x:249#y:49, decval=1492.21522
```

A subsequent invocation of `ippsolve` to the same output file would simply append the results to the end of the file:

```
$ ippsolve file2.ipp file.isr
file2.ipp: Building...  Done. Solving... Done.
$ cat file.isr
ipp=file.ipp,  cpu_time=0.34, dim=2, isol=false, ipfs=2, alg=0, pcs=100, \
               leafs=6, thresh=100, decision=x:249#y:49, decval=1492.21522
ipp=file2.ipp, cpu_time=0.41, dim=2, isol=false, ipfs=2, alg=0, pcs=100, \
               leafs=4, thresh=100, decision=x:399#y:449, decval=1403.666
```

Each result line contains all the elements of interest to us, the solve time, number of dimensions and IvP functions and pieces for each problem, the algorithm used for solving, whether or not an initial solution was used, the number of leafs visited during the branch and bound search, the solution threshold, the actual maximum point in the decision space, and the total utility of that point.

2.2.2 Specifying the Solution Algorithm used by ippsolve

There are four solution algorithms available, versions 1, 2, 3, and 4. The fourth is the default version as it it the fastest. For legacy reasons it is also reported in the solution files as algorithm 0, as in `alg=0` from the above example. On the command line:

```
$ ippsolve file.ipp file.isr --v1
$ ippsolve file.ipp file.isr --v2
$ ippsolve file.ipp file.isr --v3
$ ippsolve file.ipp file.isr --v4   // the default
```

Note in this paper, a "version 5" is also reported. This is simply the version 4 algorith, using an initial solution with the `--isol` parameter described below in Section 2.2.4.

2.2.3 Specifying the Solution Threshold used by ippsolve

Version 4 of the solution algorithm also makes use of a *solution threshold*. This parameter which ranges from $[0, 100]$ is by default 100, meaning the absolute global maximum solution is desired. A value of say 90 means that any solution within 10 percent of the global maximum is acceptable. This threshold is applied at the moment the upper bound estimate is applied to sub-nodes during the branch-and-bound search.

For example, if the threshold is 100, and the current best-so-far solution is 45, if the upper-bound calculation for all sub-nodes is less than 45, those sub-nodes would not be explored. If the threshold were instead 90, then values less than 50 would result in pruning instead. On the command line the `--thresh` option is used:

```
$ ippsolve file.ipp file.isr --thresh=90
$ ippsolve file.ipp file.isr --thresh=100   // the default
```

Again, the threshold is only used in the version 4 algorithm. If the threshold is provided while also selecting another version of the solution algorithm, the threshold will simply be ignored.

2.2.4 Using an Initial Solution in Invocations of ippsolve

The IvP solution algorithms, version 3 and 4, utilize pruning based on the current best-so-far solution and an upper-bound on the current set of sub-nodes. They therefore both benefit from using an initial solution, calculated before the full search begins. The initial solution heuristic employed is simply to visit each point in the decision space corresponding to the maximum point of each of the objective functions. This heuristic can be turned on, from the command line with:

```
$ ippsolve file.ipp file.isr --isol
```

The results (`.isr`) file will contain a either `isol=true` or `isol=false` for each reported problem solution.

2.2.5 Solving Simple Solutions Multiple Times For Timing

Some problems instances and some solution algorithms lend themselves to solution times so small that they are hard to measure. The timing mechanism used in `ippsolve` rounds to the nearest 100th of a second. To get better accuracy, the `--cnt=N` parameter may be used which will solve the given problem N times and devide by N to get the average result. This is evoked from the command line with:

```
$ ippsolve file.ipp file.isr --cnt=20
```

There is no indication in the solution output that this parameter was utilized.

2.3 The ippview Utility for Rendering IvP Problems

The `ippview` utility allows the user to visually inspect IvP problems in two dimensions. It accepts as command line input a filename containing an IvP problem instance (an `.ipp` file):

```
$ ippview file.ipp
```

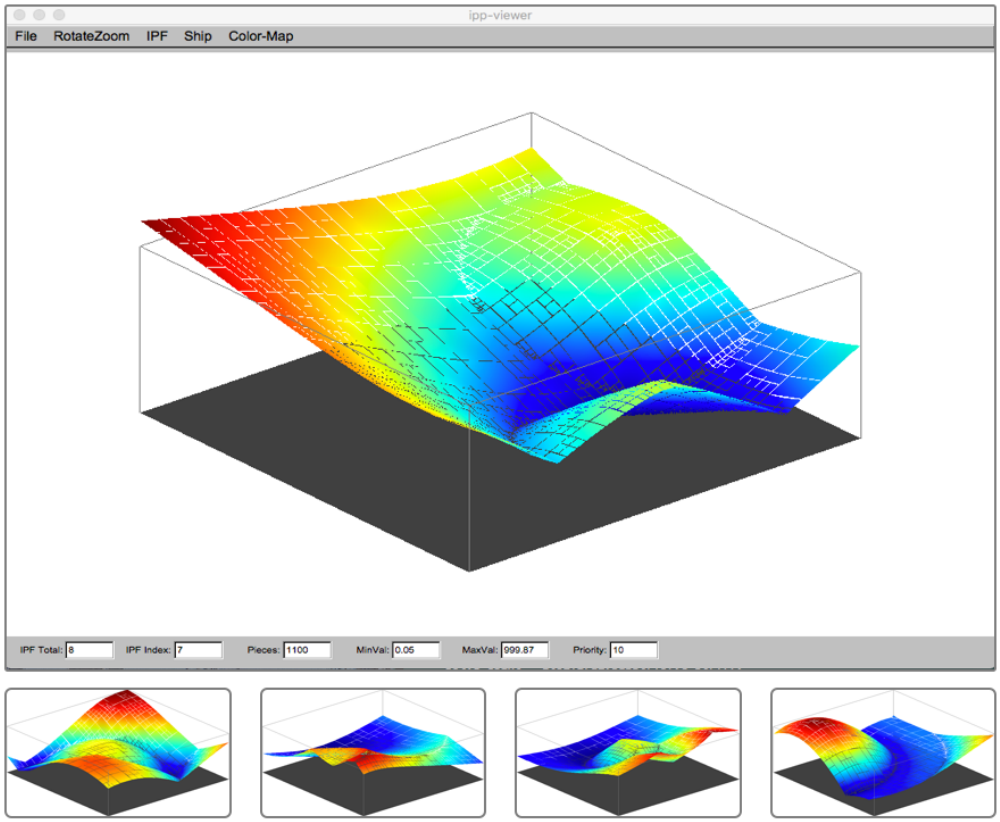

Figure 9: The `ippview` utility allows the user to visualize two-dimensional problems, inspecting each generated objective function.

2.4 The isrplot Utility for Generating Plots from Results

The `isrplot` utility allows the user to generate files suitable for making plots with GNUPlot, given a results (`.isr` file) as input. For example, the below command created the file for GNUPlot that created Figure 14 on Page 33.

```
$ isrplot test_results_v3.isr test_results_v3.dat -f --x=pcs --style=whiskers
```

In this case the below file was created, with the x-axis in column 1 being the number of pieces in each objective function. Column 4 is the average solve time. Columns 2 and 6 are the min and max solve times, and columns 3 and 5 are the standard deviation.

```
//-------------------------------------------------------------
// test_results_v3.dat file generated by the isrplot utility
//-------------------------------------------------------------
1000 0.000000 0.007640 0.009600 0.011560 0.010000
2000 0.020000 0.020000 0.024000 0.028899 0.030000
3000 0.040000 0.040000 0.040800 0.043513 0.050000
4000 0.050000 0.051101 0.056000 0.060899 0.060000
5000 0.070000 0.070000 0.074400 0.079364 0.080000
6000 0.090000 0.090000 0.093200 0.097865 0.100000
7000 0.100000 0.109225 0.115600 0.121975 0.130000
8000 0.120000 0.131528 0.141600 0.151672 0.170000
9000 0.150000 0.155862 0.166400 0.176938 0.190000
10000 0.170000 0.178203 0.185200 0.192197 0.200000
12000 0.220000 0.231056 0.240000 0.248944 0.260000
14000 0.280000 0.289046 0.300000 0.310954 0.320000
16000 0.340000 0.350376 0.366800 0.383224 0.410000
18000 0.410000 0.435785 0.455600 0.475415 0.500000
20000 0.490000 0.512524 0.541600 0.570676 0.610000
25000 0.750000 0.787346 0.828400 0.869454 0.910000
30000 1.040000 1.108847 1.184000 1.259153 1.340000
```

The plot in Figure 14 is made from the resulting `test_results_v3.dat` file using the below GNUPlot command.

```
$ plot 'test_results_v3.dat'  using 1:4 with boxes fs solid 0.55
```

2.5 The isrtex Utility for Generating LaTex Tables from Results

The `isrtex` utility is similar to the `isrplot` utility in that it ingests a results (`.isr`) file, and produces in this case a file containing a LaTex table. For example, the below command, was used to generate Table 9 on Page 34:

```
$ isrtex test_results_v3.isr --tname=ExpThreeTwoDTenIpfsVThree latex_table_s3a_v3.tex
```

The LaTex output is shown below:

```
%=============================================
% ExpThreeTwoDTenIpfsVThree
%=============================================
\newcommand{\ExpThreeTwoDTenIpfsVThree}{
\footnotesize
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}  \hline
&  \multicolumn{8}{|c|}{dim=2, ipfs=10} \\ \cline{2-9}
&  \multicolumn{4}{|c|}{CPU Time (seconds)} &    \multicolumn{4}{|c|}{Leaf Nodes} \\
\cline{2-9} \hline
pcs & avg & sig & min & max & avg & sig & min & max \\ \hline
\hline
 1000 & 0.010 & 0.002 & 0.00 & 0.01 & 3731.20 & 61.51 & 3619 & 3854 \\
 2000 & 0.024 & 0.005 & 0.02 & 0.03 & 10751.32 & 217.13 & 10172 & 11154 \\
 3000 & 0.041 & 0.003 & 0.04 & 0.05 & 16933.52 & 349.93 & 16155 & 17567 \\
 4000 & 0.056 & 0.005 & 0.05 & 0.06 & 22200.84 & 509.41 & 21121 & 23113 \\
 5000 & 0.074 & 0.005 & 0.07 & 0.08 & 27337.28 & 897.22 & 25393 & 28831 \\
 6000 & 0.093 & 0.005 & 0.09 & 0.10 & 31866.36 & 1022.59 & 29468 & 33844 \\
 7000 & 0.116 & 0.006 & 0.10 & 0.13 & 36199.76 & 1335.38 & 33558 & 39818 \\
 8000 & 0.142 & 0.010 & 0.12 & 0.17 & 39678.80 & 1014.28 & 37267 & 41075 \\
 9000 & 0.166 & 0.011 & 0.15 & 0.19 & 43937.84 & 1324.59 & 40675 & 46008 \\
10000 & 0.185 & 0.007 & 0.17 & 0.20 & 47991.28 & 1533.83 & 43622 & 50452 \\
12000 & 0.240 & 0.009 & 0.22 & 0.26 & 54107.72 & 1877.98 & 49559 & 58054 \\
14000 & 0.300 & 0.011 & 0.28 & 0.32 & 60221.24 & 1907.67 & 57270 & 64658 \\
16000 & 0.367 & 0.016 & 0.34 & 0.41 & 66409.28 & 2463.15 & 60995 & 72427 \\
18000 & 0.456 & 0.020 & 0.41 & 0.50 & 72117.52 & 2354.33 & 66293 & 76262 \\
20000 & 0.542 & 0.029 & 0.49 & 0.61 & 77570.96 & 3271.35 & 69212 & 82310 \\
25000 & 0.828 & 0.041 & 0.75 & 0.91 & 91039.16 & 3409.73 & 84451 & 96665 \\
30000 & 1.184 & 0.075 & 1.04 & 1.34 & 104199.32 & 3475.84 & 97690 & 111401 \\
\hline
\end{tabular}
\normalsize}
```

3 Trial 1: RIPAL Version 1 vs. Version 2

Trial 1 contains a pair of trials, 1A and 1B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 1 and Version 2 are compared. The details of these algorithms are outside the scope here. In short RIPAL Version 1 is the full expansion of the IvP search tree with no pruning. Version 2 prunes the tree when an infeasible node is encountered, meaning the set of pieces at that node do not intersect. Version 1 is not realistically ever used, but is interesting only since it represents a full expansion of the search tree. The plotted run times are so different that two different y (time) axes are used on each plot.

3.1 Trial 1A: RIPAL Version 1 vs. Version 2 (3 IvP Functions)

In Trial 1A, each IvP problem has two dimensions and three IvP functions. The number of pieces is varied between 400 and 1000 pieces. For each problem type, 25 random instances were generated.

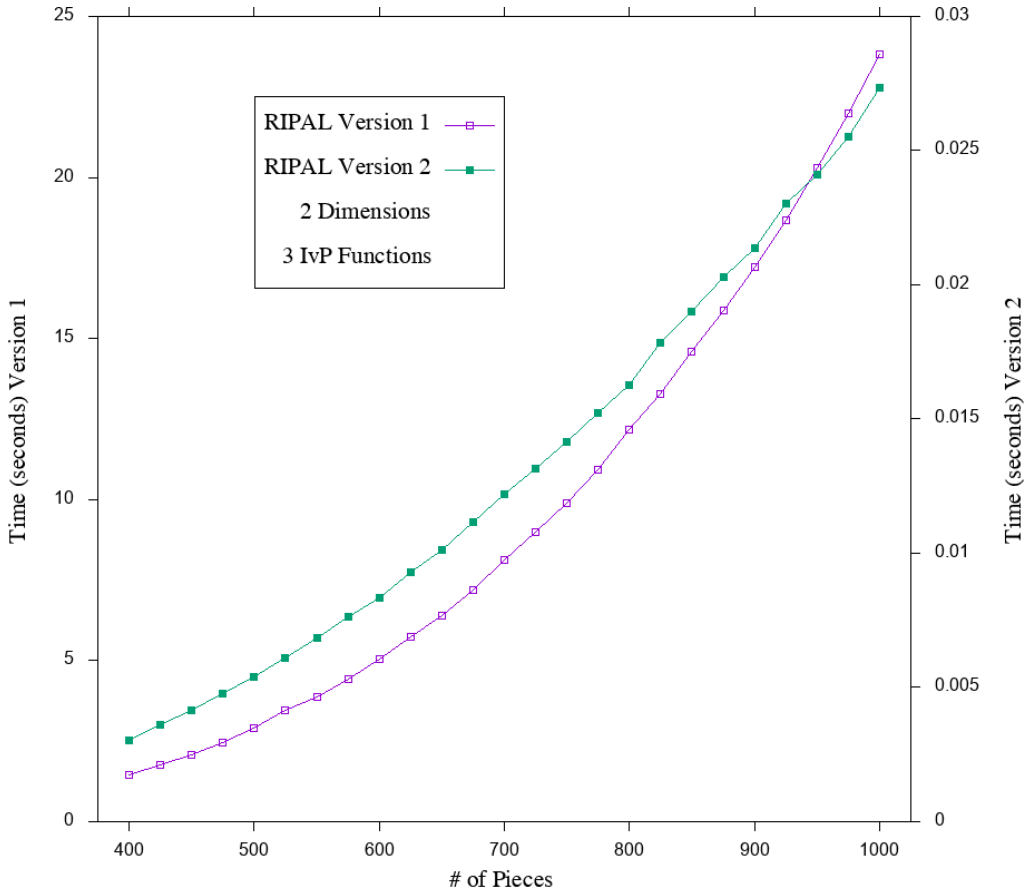

Figure 10: Trial 1A RIPAL: Version 1 vs. Version 2. Note the separate y (time) axes for each algorithm.

3.1.1 Data From Trial A - RIPAL Version 1

Table 1 contains the full results of RIPAL Version 1, for 25 randomly generated IvP problems, plotted in Figure 10. Time is shown in seconds.

	dim=2, ipfs=3							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
400	1.456	0.006	1.45	1.47	64000000.00	0.00	64000000	64000000
425	1.758	0.006	1.75	1.77	76765625.00	0.00	76765625	76765625
450	2.090	0.008	2.07	2.10	91125000.00	0.00	91125000	91125000
475	2.468	0.015	2.44	2.49	107171875.00	0.00	107171875	107171875
500	2.891	0.027	2.85	2.98	125000000.00	0.00	125000000	125000000
525	3.456	0.073	3.29	3.60	144703125.00	0.00	144703125	144703125
550	3.872	0.036	3.80	3.93	166375000.00	0.00	166375000	166375000
575	4.440	0.048	4.35	4.55	190109375.00	0.00	190109375	190109375
600	5.050	0.068	4.95	5.32	216000000.00	0.00	216000000	216000000
625	5.740	0.071	5.56	5.85	244140625.00	0.00	244140625	244140625
650	6.400	0.095	6.19	6.57	274625000.00	0.00	274625000	274625000
675	7.203	0.132	6.94	7.42	307546875.00	0.00	307546875	307546875
700	8.117	0.146	7.79	8.44	343000000.00	0.00	343000000	343000000
725	8.975	0.136	8.67	9.22	381078125.00	0.00	381078125	381078125
750	9.898	0.172	9.67	10.35	421875000.00	0.00	421875000	421875000
775	10.922	0.167	10.69	11.23	465484375.00	0.00	465484375	465484375
800	12.168	0.221	11.82	12.64	512000000.00	0.00	512000000	512000000
825	13.284	0.199	12.71	13.60	561515625.00	0.00	561515625	561515625
850	14.588	0.339	13.80	15.25	614125000.00	0.00	614125000	614125000
875	15.880	0.257	15.31	16.48	669921875.00	0.00	669921875	669921875
900	17.224	0.365	16.68	17.89	729000000.00	0.00	729000000	729000000
925	18.685	0.425	17.76	19.32	791453125.00	0.00	791453125	791453125
950	20.306	0.409	19.30	21.00	857375000.00	0.00	857375000	857375000
975	21.988	0.460	20.84	22.85	926859375.00	0.00	926859375	926859375
1000	23.824	0.535	22.86	24.66	1000000000.00	0.00	1000000000	1000000000

Table 1: Trial 1A RIPAL Version 1: All problems have 2 dimensions, 3 IvP functions. Solution time, in seconds, is explored as the number of pieces vary.

3.1.2 Data From Trial 1A - RIPAL Version 2

Table 2 contains the full results of RIPAL Version 2, for 25 randomly generated IvP problems, plotted in Figure 10. Time is shown in seconds.

	dim=2, ipfs=3							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
400	0.003	0.000	0.00	0.00	200000.00	0.00	200000	200000
425	0.004	0.000	0.00	0.00	236540.00	947.84	234000	237500
450	0.004	0.000	0.00	0.00	272780.00	1805.99	268500	275000
475	0.005	0.000	0.00	0.00	308360.00	3028.27	300500	312500
500	0.005	0.000	0.01	0.01	343420.00	2567.80	338000	347000
525	0.006	0.000	0.01	0.01	378060.00	3899.54	370000	385000
550	0.007	0.000	0.01	0.01	414460.00	4303.30	406500	423500
575	0.008	0.000	0.01	0.01	449720.00	5105.06	440000	459500
600	0.008	0.000	0.01	0.01	484060.00	7920.00	467500	498000
625	0.009	0.000	0.01	0.01	517480.00	6032.38	503500	528000
650	0.010	0.000	0.01	0.01	556520.00	8211.55	542000	575000
675	0.011	0.000	0.01	0.01	588840.00	9494.97	565000	605500
700	0.012	0.000	0.01	0.01	623260.00	11069.89	586000	638500
725	0.013	0.000	0.01	0.01	657200.00	14023.55	628500	677000
750	0.014	0.000	0.01	0.02	690000.00	11618.95	661000	705000
775	0.015	0.000	0.01	0.02	718800.00	10280.08	693000	736500
800	0.016	0.000	0.02	0.02	756460.00	15228.87	726500	782500
825	0.018	0.000	0.02	0.02	794440.00	15265.20	761500	834500
850	0.019	0.001	0.02	0.02	826520.00	15857.79	776000	858000
875	0.020	0.001	0.02	0.02	857320.00	15601.21	831000	884000
900	0.021	0.001	0.02	0.02	893180.00	14965.21	862000	918500
925	0.023	0.001	0.02	0.02	916500.00	11890.33	894000	942500
950	0.024	0.001	0.02	0.03	956160.00	22483.20	906500	1009000
975	0.025	0.001	0.02	0.03	985020.00	21776.35	941000	1024500
1000	0.027	0.001	0.03	0.03	1027320.00	14029.17	991500	1047500

Table 2: Trial 1A RIPAL Version 2: All problems have 2 dimensions, 3 IvP functions. Solution time, in seconds, is explored as the number of pieces vary.

3.2 Trial 1B: RIPAL Version 1 vs. Version 2 (4 IvP Functions)

In Trial 1B, each IvP problem has two dimensions and four IvP functions. The number of pieces is varied between 100 and 500 pieces. For each problem type, 25 random instances were generated.

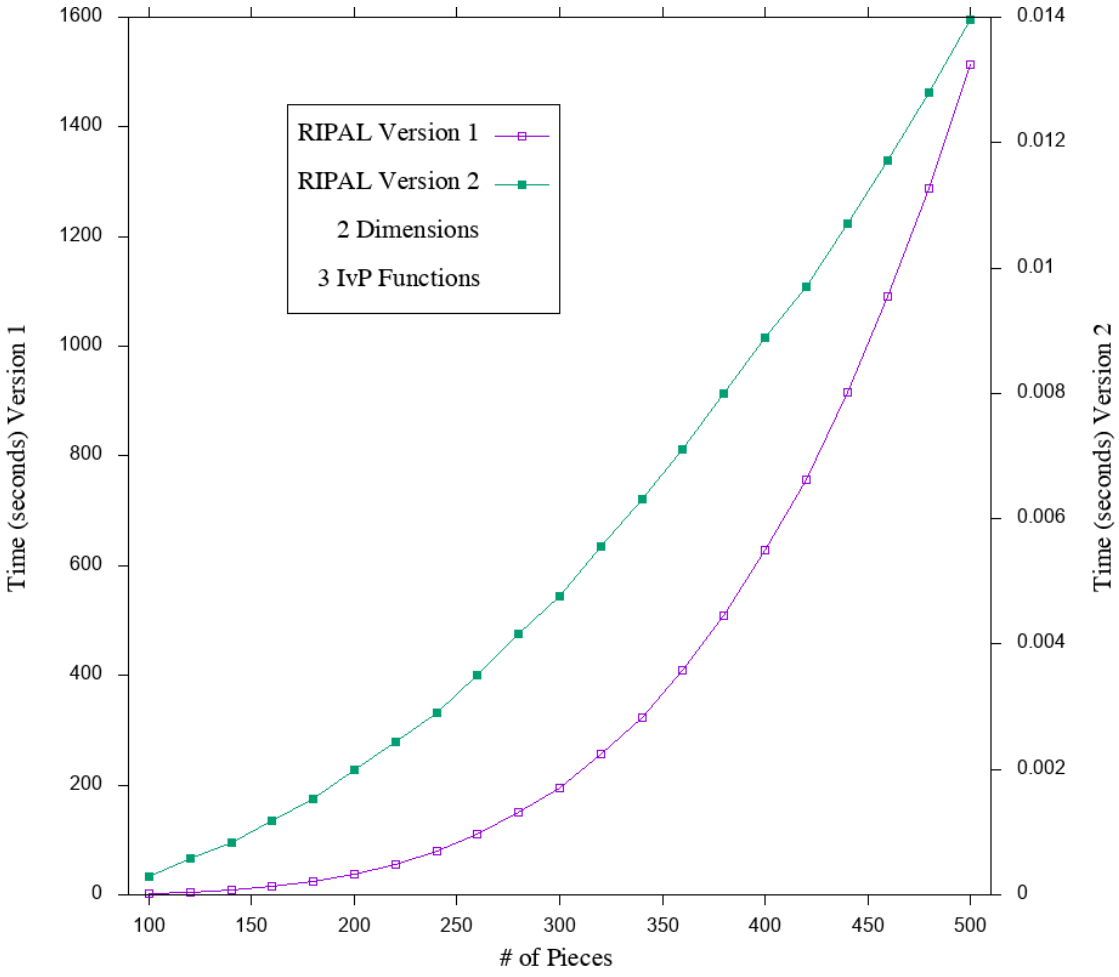

Figure 11: Comparison of RIPAL: version-1 vs. version-2. Note the separate y (time) axes for each algorithm.

3.2.1 Data From Trial 1B - RIPAL Version 1

Table 3 contains the full results of RIPAL Version 1, for 25 randomly generated IvP problems, with 2 dimensions and 4 IvP functions each, plotted in Figure 11. Time is shown in seconds.

	dim=2, ipfs=4							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
100	2.388	0.019	2.37	2.42	100000000.00	0.00	100000000	100000000
120	4.990	0.074	4.90	5.21	207360000.00	0.00	207360000	207360000
140	9.223	0.101	9.05	9.43	384160000.00	0.00	384160000	384160000
160	15.826	0.348	15.17	16.61	655360000.00	0.00	655360000	655360000
180	25.159	0.488	24.17	26.21	1049760000.00	0.00	1049760000	1049760000
200	38.547	0.828	37.17	40.12	1600000000.00	0.00	1600000000	1600000000
220	56.128	1.419	53.92	58.62	2342560000.00	0.00	2342560000	2342560000
240	79.334	1.886	76.07	82.07	3317760000.00	0.00	3317760000	3317760000
260	109.724	2.672	104.53	114.76	274792704.00	0.00	274792704	274792704
280	149.609	2.145	144.52	153.88	1851592704.00	0.00	1851592704	1851592704
300	194.075	6.716	184.22	205.39	3805032704.00	0.00	3805032704	3805032704
320	257.065	8.520	244.67	280.21	1895825408.00	0.00	1895825408	1895825408
340	323.587	11.006	300.93	339.43	478458112.00	0.00	478458112	478458112
360	409.706	12.075	381.53	429.02	3911258112.00	0.00	3911258112	3911258112
380	510.044	13.589	489.86	531.63	3671490816.00	0.00	3671490816	3671490816
400	629.341	13.523	608.13	656.30	4125163520.00	0.00	4125163520	4125163520
420	757.231	21.447	704.20	801.54	1052188928.00	0.00	1052188928	1052188928
440	916.949	24.425	869.94	953.45	3121221632.00	0.00	3121221632	3121221632
460	1091.593	28.970	1028.02	1148.16	1824887040.00	0.00	1824887040	1824887040
480	1288.022	62.923	1171.62	1391.63	1544552448.00	0.00	1544552448	1544552448
500	1514.626	58.678	1395.78	1638.33	2370457856.00	0.00	2370457856	2370457856

Table 3: All problems have 2 dimensions, 4 IvP functions. Solution time is explored as the number of pieces vary. The solution algorithm is RIPAL Version 1.

3.2.2 Data From Trial 1B - RIPAL Version 2

Table 4 contains the full results of RIPAL Version 2, for 25 randomly generated IvP problems, with 2 dimensions and 4 IvP functions each, plotted in Figure 11. Time is shown in seconds.

	dim=2, ipfs=4							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
100	0.000	0.000	0.00	0.00	50000.00	0.00	50000	50000
120	0.001	0.000	0.00	0.00	86600.00	1479.86	84500	90000
140	0.001	0.000	0.00	0.00	120775.00	2517.32	114500	124500
160	0.001	0.000	0.00	0.00	156625.00	4095.35	148500	163500
180	0.002	0.000	0.00	0.00	185950.00	5150.00	177000	195500
200	0.002	0.000	0.00	0.00	218775.00	5510.16	206500	231000
220	0.002	0.000	0.00	0.00	253700.00	8133.88	236000	271500
240	0.003	0.000	0.00	0.00	282350.00	12198.46	251000	307000
260	0.004	0.000	0.00	0.00	317250.00	9130.31	300000	336000
280	0.004	0.000	0.00	0.00	348150.00	8756.57	337500	370500
300	0.005	0.000	0.00	0.01	377775.00	12195.77	360000	397000
320	0.006	0.000	0.01	0.01	408050.00	13127.17	382000	437500
340	0.006	0.000	0.01	0.01	446525.00	17209.35	410000	472000
360	0.007	0.000	0.01	0.01	471575.00	16501.72	436000	508000
380	0.008	0.000	0.01	0.01	505550.00	14626.09	485000	537500
400	0.009	0.000	0.01	0.01	529425.00	24227.71	486000	588500
420	0.010	0.000	0.01	0.01	561350.00	17898.39	527000	596000
440	0.011	0.000	0.01	0.01	594800.00	15624.02	553000	616500
460	0.012	0.000	0.01	0.01	623450.00	21867.16	570000	663500
480	0.013	0.001	0.01	0.01	662325.00	22194.75	626500	707000
500	0.014	0.001	0.01	0.01	690525.00	20900.64	644500	729000

Table 4: All problems have 2 dimensions, 4 IvP functions. Solution time is explored as the number of pieces vary. The solution algorithm is RIPAL Version 2.

4 Trial 2: RIPAL Version 2 vs. Version 3

Trial 2 contains a pair of trials, 2A and 2B. In each trial the the Recursive Interval Programming Algorithm (RIPAL) Version 2 and Version 3 are compared. The details of these algorithms are outside the scope here. In short RIPAL Version 2 prunes the tree when an infeasible node is encountered, meaning the set of pieces at that node do not intersect. Version 3 uses a grid structure to facilitate the finding of pieces in functions below the current search node that intersect the current node. Version 2 is not realistically ever used, but is interesting only since it represents a very simple method of pruning the search tree. The plotted run times are so different that two different y (time) axes are used on each plot.

4.1 Trial 2A: RIPAL Version 2 vs. Version 3 (2 IvP Functions)

In Trial 2A, each IvP problem has two dimensions and two IvP functions. The number of pieces is varied between 1000 and 20000 pieces. For each problem type, 25 random instances were generated.

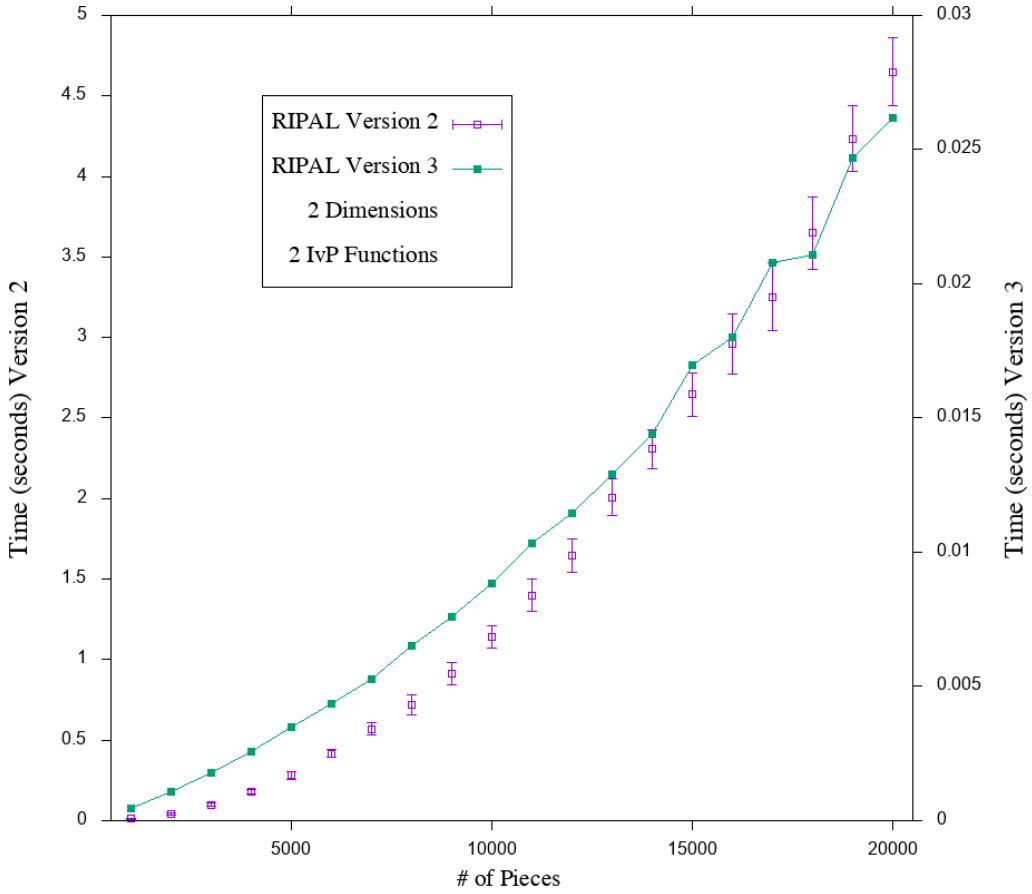

Figure 12: Trial 2A RIPAL: Version 2 vs. Version 3. Note the separate y (time) axes for each algorithm.

4.1.1 Data From Trial 2A - RIPAL Version 2

Table 5 contains the full results of RIPAL Version 2, for 25 randomly generated IvP problems, plotted in Figure 12. Time is shown in seconds.

	dim=2, ipfs=2							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.010	0.002	0.01	0.02	1357.52	16.13	1304	1375
2000	0.043	0.005	0.03	0.05	3253.32	73.69	3076	3356
3000	0.098	0.009	0.08	0.12	5043.68	104.14	4882	5255
4000	0.181	0.013	0.15	0.20	6729.28	222.59	6354	7146
5000	0.282	0.023	0.24	0.32	8341.56	355.64	7556	9125
6000	0.417	0.026	0.38	0.47	9966.88	353.94	9325	10834
7000	0.570	0.038	0.46	0.63	11699.88	426.19	10751	12451
8000	0.718	0.060	0.60	0.83	13253.84	616.09	12149	14470
9000	0.913	0.068	0.75	1.02	14845.24	760.13	13414	16258
10000	1.141	0.068	1.00	1.27	16442.00	596.45	15317	17543
11000	1.400	0.101	1.13	1.52	17839.84	972.24	16023	20112
12000	1.648	0.104	1.44	1.84	19534.56	843.19	17108	21621
13000	2.008	0.113	1.75	2.17	20964.08	993.87	18808	22168
14000	2.307	0.122	2.02	2.60	22575.36	851.41	20865	24202
15000	2.646	0.138	2.34	2.88	23768.72	1344.46	20465	26056
16000	2.958	0.186	2.58	3.28	25268.00	1168.92	23629	28737
17000	3.252	0.210	2.76	3.61	27031.92	1311.84	24412	29919
18000	3.650	0.224	3.20	4.04	28914.24	1638.57	26174	32824
19000	4.234	0.203	3.69	4.56	29209.04	1474.50	25039	31015
20000	4.648	0.210	4.24	4.98	31081.40	1421.75	28966	34231

Table 5: Trial 2A RIPAL Version 2: All problems have 2 dimensions, 2 IvP functions. Solution time is explored as the number of pieces vary.

4.1.2 Data From Trial 2A - RIPAL Version 3

Table 6 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 12. Time is shown in seconds.

	dim=2, ipfs=2							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.000	0.000	0.00	0.00	135752.00	1613.47	130400	137500
2000	0.001	0.000	0.00	0.00	325332.00	7369.00	307600	335600
3000	0.002	0.000	0.00	0.00	504368.00	10413.75	488200	525500
4000	0.003	0.000	0.00	0.00	672928.00	22258.63	635400	714600
5000	0.003	0.000	0.00	0.00	834156.00	35564.31	755600	912500
6000	0.004	0.000	0.00	0.01	996688.00	35393.97	932500	1083400
7000	0.005	0.000	0.00	0.01	1169988.00	42618.92	1075100	1245100
8000	0.007	0.001	0.01	0.01	1325384.00	61608.73	1214900	1447000
9000	0.008	0.001	0.01	0.01	1484524.00	76013.45	1341400	1625800
10000	0.009	0.001	0.01	0.01	1644200.00	59645.31	1531700	1754300
11000	0.010	0.001	0.01	0.01	1783984.00	97224.21	1602300	2011200
12000	0.011	0.001	0.01	0.01	1953456.00	84318.77	1710800	2162100
13000	0.013	0.001	0.01	0.02	2096408.00	99387.21	1880800	2216800
14000	0.014	0.001	0.01	0.02	2257536.00	85141.13	2086500	2420200
15000	0.017	0.003	0.01	0.03	2376872.00	134445.96	2046500	2605600
16000	0.018	0.002	0.01	0.02	2526800.00	116892.23	2362900	2873700
17000	0.021	0.002	0.02	0.03	2703192.00	131184.43	2441200	2991900
18000	0.021	0.003	0.02	0.03	2891424.00	163857.09	2617400	3282400
19000	0.025	0.004	0.02	0.04	2920904.00	147450.31	2503900	3101500
20000	0.026	0.003	0.02	0.03	3108140.00	142174.55	2896600	3423100

Table 6: Trial 2B RIPAL Version 3: All problems have 2 dimensions, 2 IvP functions. Solution time is explored as the number of pieces vary.

4.2 Trial 2B: RIPAL Version 2 vs. Version 3 (3 IvP Functions)

In Trial 2B, each IvP problem has two dimensions and three IvP functions. The number of pieces is varied between 1000 and 20000 pieces. For each problem type, 25 random instances were generated.

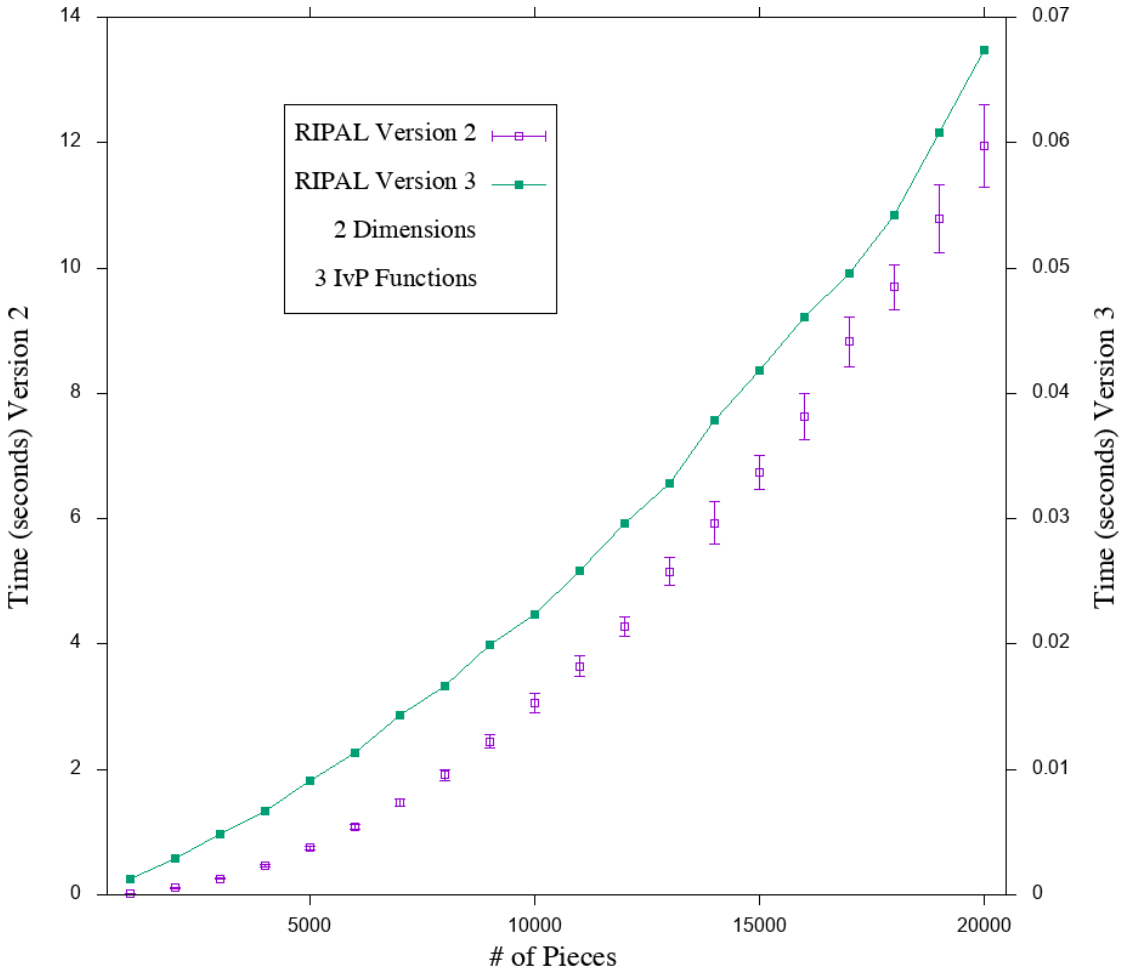

Figure 13: Trial 2B RIPAL: Version 2 vs. Version 3. Note the separate y (time) axes for each algorithm.

4.2.1 Data From Trial 2B - RIPAL Version 2

Table 7 contains the full results of RIPAL Version 2, for 25 randomly generated IvP problems, plotted in Figure 13. Time is shown in seconds.

	dim=2, ipfs=3							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.023	0.005	0.02	0.03	1695.84	27.29	1625	1736
2000	0.111	0.006	0.10	0.12	4351.24	90.96	4138	4494
3000	0.261	0.013	0.24	0.29	6807.72	210.67	6328	7087
4000	0.466	0.020	0.42	0.50	9123.92	363.62	8301	10014
5000	0.749	0.027	0.71	0.83	11330.96	422.00	9768	11998
6000	1.084	0.040	0.98	1.15	13615.20	417.49	12828	14806
7000	1.475	0.064	1.33	1.58	15514.48	403.80	14615	16180
8000	1.910	0.092	1.73	2.10	17679.04	825.07	15809	19278
9000	2.449	0.113	2.20	2.72	19613.96	883.85	18014	21725
10000	3.056	0.161	2.72	3.32	21626.80	757.67	20321	23299
11000	3.644	0.166	3.24	4.05	23323.12	1168.51	21152	26405
12000	4.276	0.152	4.00	4.63	25069.04	1036.75	22758	27624
13000	5.158	0.219	4.57	5.62	27583.52	1495.91	24795	30878
14000	5.932	0.333	5.08	6.56	29089.36	1741.22	25592	32854
15000	6.735	0.271	6.07	7.23	30894.56	1544.62	28439	33374
16000	7.629	0.371	6.98	8.28	32733.76	1976.39	29134	38122
17000	8.826	0.398	8.17	9.82	34631.68	1925.32	30726	39110
18000	9.695	0.356	8.56	10.38	36920.16	2205.61	32846	42120
19000	10.784	0.538	9.89	11.84	38909.92	2499.87	35541	46885
20000	11.956	0.659	10.48	13.49	38829.16	2301.86	34711	43239

Table 7: Trial 2B RIPAL Version 2: All problems have 2 dimensions, 3 IvP functions. Solution time is explored as the number of pieces vary.

4.2.2 Data From Trial 2B - RIPAL Version 3

Table 7 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 13. Time is shown in seconds.

	dim=2, ipfs=3							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.001	0.000	0.00	0.00	678336.00	10915.64	650000	694400
2000	0.003	0.000	0.00	0.00	1740496.00	36383.61	1655200	1797600
3000	0.005	0.000	0.00	0.01	2723088.00	84267.04	2531200	2834800
4000	0.007	0.000	0.01	0.01	3649568.00	145447.04	3320400	4005600
5000	0.009	0.000	0.01	0.01	4532384.00	168801.23	3907200	4799200
6000	0.011	0.000	0.01	0.01	5446080.00	166995.17	5131200	5922400
7000	0.014	0.000	0.01	0.02	6205792.00	161520.10	5846000	6472000
8000	0.017	0.001	0.02	0.02	7071616.00	330027.56	6323600	7711200
9000	0.020	0.001	0.02	0.02	7845584.00	353541.40	7205600	8690000
10000	0.022	0.001	0.02	0.02	8650720.00	303066.21	8128400	9319600
11000	0.026	0.001	0.02	0.03	9329248.00	467404.33	8460800	10562000
12000	0.030	0.002	0.03	0.04	10027616.00	414700.18	9103200	11049600
13000	0.033	0.002	0.03	0.04	11033408.00	598362.85	9918000	12351200
14000	0.038	0.003	0.03	0.05	11635744.00	696489.55	10236800	13141600
15000	0.042	0.002	0.04	0.05	12357824.00	617847.34	11375600	13349600
16000	0.046	0.004	0.04	0.06	13093504.00	790555.43	11653600	15248800
17000	0.050	0.003	0.04	0.06	13852672.00	770129.87	12290400	15644000
18000	0.054	0.003	0.05	0.06	14768064.00	882242.14	13138400	16848000
19000	0.061	0.006	0.05	0.08	15563968.00	999948.42	14216400	18754000
20000	0.067	0.007	0.06	0.09	15531664.00	920742.81	13884400	17295600

Table 8: Trial 2B `RIPAL`Version 3: All problems have 2 dimensions, 3 IvP functions. Solution time is explored as the number of pieces vary.

5 Trial 3: RIPAL Version 3 vs. Version 4 (10 IvP Functions)

Trial 3 contains a pair of trials, 3A and 3B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 3 and Version 4 are compared. The details of these algorithms are outside the scope here. In short, RIPAL Version 3 uses a grid structure to facilitate the finding of pieces in functions below the current search node that intersect the current node. Version 4 also uses this grid structure to calculate an upper bound on solutions below the current search node. Version 3 is rarely used in practice since using an upper bound almost always speeds up the solution. In some cases, with many IvP functions and thus a very deep search tree, the upper bound calculation is counter-productive and Version 3 is actually faster (See Figure 19).

5.1 Trial 3A: RIPAL Version 3 vs. Version 4 (2 Dimensions)

In Trial 3A, each IvP problem has two dimensions and ten IvP functions. The number of pieces is varied between 1000 and 30000 pieces. For each problem type, 25 random instances were generated.

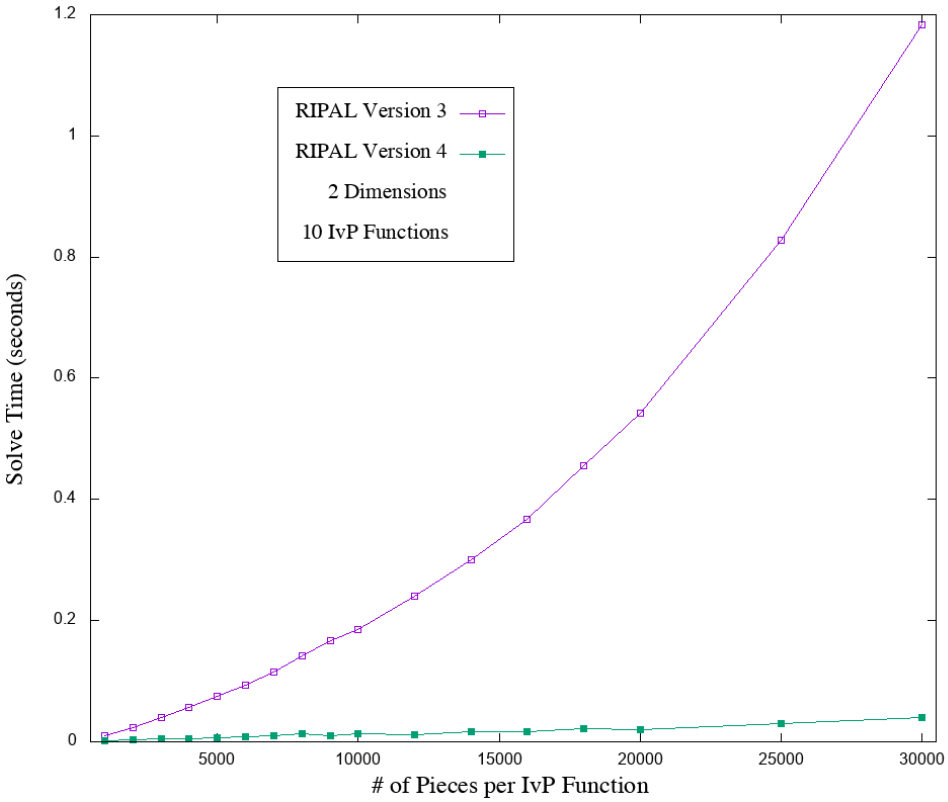

Figure 14: Trial 3A RIPAL: Version 3 vs. Version 4.

5.1.1 Data From Trial 3A - RIPAL Version 3

Table 9 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 14. Time is shown in seconds.

	dim=2, ipfs=10							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.010	0.002	0.00	0.01	3731.20	61.51	3619	3854
2000	0.024	0.005	0.02	0.03	10751.32	217.13	10172	11154
3000	0.041	0.003	0.04	0.05	16933.52	349.93	16155	17567
4000	0.056	0.005	0.05	0.06	22200.84	509.41	21121	23113
5000	0.074	0.005	0.07	0.08	27337.28	897.22	25393	28831
6000	0.093	0.005	0.09	0.10	31866.36	1022.59	29468	33844
7000	0.116	0.006	0.10	0.13	36199.76	1335.38	33558	39818
8000	0.142	0.010	0.12	0.17	39678.80	1014.28	37267	41075
9000	0.166	0.011	0.15	0.19	43937.84	1324.59	40675	46008
10000	0.185	0.007	0.17	0.20	47991.28	1533.83	43622	50452
12000	0.240	0.009	0.22	0.26	54107.72	1877.98	49559	58054
14000	0.300	0.011	0.28	0.32	60221.24	1907.67	57270	64658
16000	0.367	0.016	0.34	0.41	66409.28	2463.15	60995	72427
18000	0.456	0.020	0.41	0.50	72117.52	2354.33	66293	76262
20000	0.542	0.029	0.49	0.61	77570.96	3271.35	69212	82310
25000	0.828	0.041	0.75	0.91	91039.16	3409.73	84451	96665
30000	1.184	0.075	1.04	1.34	104199.32	3475.84	97690	111401

Table 9: Trial 3A `RIPAL` Version 3: All problems have 2 dimensions, 10 IvP functions. Solution time is explored as the number of pieces vary.

5.1.2 Data From Trial 3A - RIPAL Version 4

Table 10 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 14. Time is shown in seconds.

	dim=2, ipfs=10							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.002	0.004	0.00	0.01	26.76	11.15	3	51
2000	0.003	0.005	0.00	0.01	34.00	14.09	4	60
3000	0.005	0.005	0.00	0.01	42.92	18.26	12	82
4000	0.005	0.005	0.00	0.01	34.36	17.79	9	86
5000	0.007	0.004	0.00	0.01	44.36	17.72	16	92
6000	0.008	0.004	0.00	0.01	40.44	11.65	19	63
7000	0.010	0.005	0.00	0.02	44.00	17.17	21	84
8000	0.013	0.004	0.01	0.02	43.48	19.06	11	103
9000	0.010	0.002	0.01	0.02	38.92	11.26	22	69
10000	0.014	0.005	0.01	0.02	40.28	16.52	11	74
12000	0.012	0.006	0.00	0.03	43.96	16.48	19	78
14000	0.017	0.007	0.01	0.03	44.32	22.90	13	97
16000	0.016	0.007	0.01	0.03	44.04	19.61	15	102
18000	0.021	0.008	0.01	0.04	37.60	13.14	15	71
20000	0.020	0.006	0.01	0.03	37.76	17.40	9	73
25000	0.030	0.011	0.01	0.05	41.04	23.43	13	103
30000	0.040	0.018	0.01	0.09	45.12	22.36	10	95

Table 10: Trial 3A `RIPAL`Version 4: All problems have 2 dimensions, 10 IvP functions. Solution time is explored as the number of pieces vary.

5.2 Trial 3B: RIPAL Version 3 vs. Version 4 (4 Dimensions)

In Trial 3B, each IvP problem has four dimensions and ten IvP functions. The number of pieces is varied between 1000 and 30000 pieces. For each problem type, 25 random instances were generated.

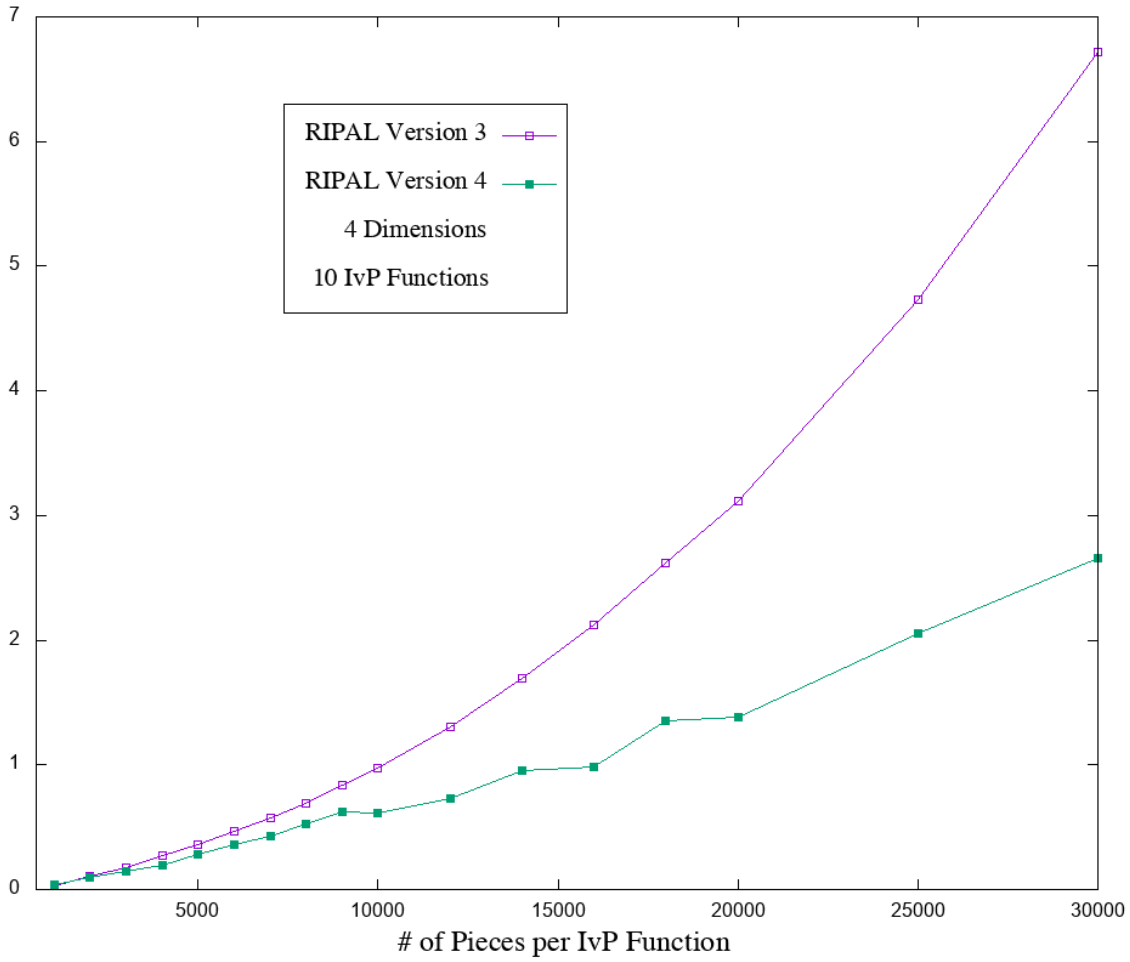

Figure 15: Trial 3B RIPAL Version 3 vs. Version 4.

5.2.1 Data From Trial 3B - RIPAL Version 3

Table 11 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 15. Time is shown in seconds.

	dim=4, ipfs=20							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.034	0.005	0.03	0.04	5777.72	205.59	5359	6114
2000	0.103	0.005	0.09	0.11	18965.72	808.77	17348	20474
3000	0.179	0.008	0.16	0.19	32383.32	1597.59	29862	35496
4000	0.268	0.008	0.26	0.29	46644.08	2008.28	41457	50205
5000	0.363	0.013	0.33	0.39	60248.68	2650.13	55617	65701
6000	0.464	0.013	0.44	0.50	73410.76	2306.18	68154	78159
7000	0.579	0.020	0.53	0.61	88630.92	3789.43	82814	97243
8000	0.694	0.024	0.66	0.75	103067.56	4038.96	95023	112068
9000	0.836	0.031	0.79	0.92	116223.16	3162.69	109253	122608
10000	0.977	0.027	0.94	1.04	132851.00	4047.46	125854	143468
12000	1.304	0.057	1.21	1.40	159785.12	5517.83	150044	172993
14000	1.690	0.085	1.54	1.85	190590.88	6177.72	180022	203503
16000	2.124	0.122	1.89	2.40	220511.32	7540.31	202754	234022
18000	2.614	0.183	2.28	3.00	252412.24	8559.09	232633	268890
20000	3.118	0.227	2.72	3.61	281705.40	8555.50	266207	298177
25000	4.735	0.289	4.19	5.63	359031.28	11550.46	338204	378086
30000	6.719	0.672	5.73	8.08	443372.20	15198.14	415857	474687

Table 11: Trial 3B `RIPAL` Version 3: All problems have 4 dimensions, 10 IvP functions. Solution time is explored as the number of pieces vary.

5.2.2 Data From Trial 3B - RIPAL Version 4

Table 12 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 15. Time is shown in seconds.

	dim=4, ipfs=20							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.035	0.013	0.01	0.06	16.48	7.47	5	32
2000	0.097	0.035	0.02	0.17	25.12	11.70	5	49
3000	0.143	0.054	0.06	0.28	28.60	7.55	10	47
4000	0.192	0.076	0.04	0.32	29.40	11.47	11	57
5000	0.278	0.104	0.10	0.49	33.24	13.34	17	73
6000	0.361	0.145	0.09	0.62	28.72	11.73	9	68
7000	0.425	0.112	0.26	0.66	29.04	10.49	10	56
8000	0.528	0.165	0.19	0.88	32.96	11.60	7	54
9000	0.620	0.208	0.29	1.05	38.08	13.48	14	64
10000	0.618	0.212	0.11	1.08	36.48	11.25	17	61
12000	0.729	0.238	0.21	1.22	28.64	11.12	12	53
14000	0.953	0.272	0.42	1.49	32.40	14.90	6	74
16000	0.986	0.401	0.29	1.91	43.40	20.34	15	104
18000	1.356	0.530	0.47	2.35	37.84	16.94	11	89
20000	1.380	0.438	0.48	2.05	43.44	12.87	26	75
25000	2.052	0.959	0.40	4.06	38.04	16.13	12	71
30000	2.661	0.966	0.44	4.63	43.04	17.21	19	80

Table 12: Trial 3B: `RIPAL` Version 4: All problems have 4 dimensions, 10 IvP functions. Solution time is explored as the number of pieces vary.

6 Trial 4: RIPAL Version 3 vs. Version 4 (20 IvP Functions)

Trial 4 contains a pair of trials, 4A and 4B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 3 and Version 4 are compared. The details of these algorithms are outside the scope here. This trial is similar to Trial 3 with the primary difference being that 20 IvP functions are used instead of 10.

6.1 Trial 4A: RIPAL Version 3 vs. Version 4 (2 Dimensions)

In Trial 4A, each IvP problem has two dimensions and twenty IvP functions. The number of pieces is varied between 1000 and 30000 pieces. For each problem type, 25 random instances were generated.

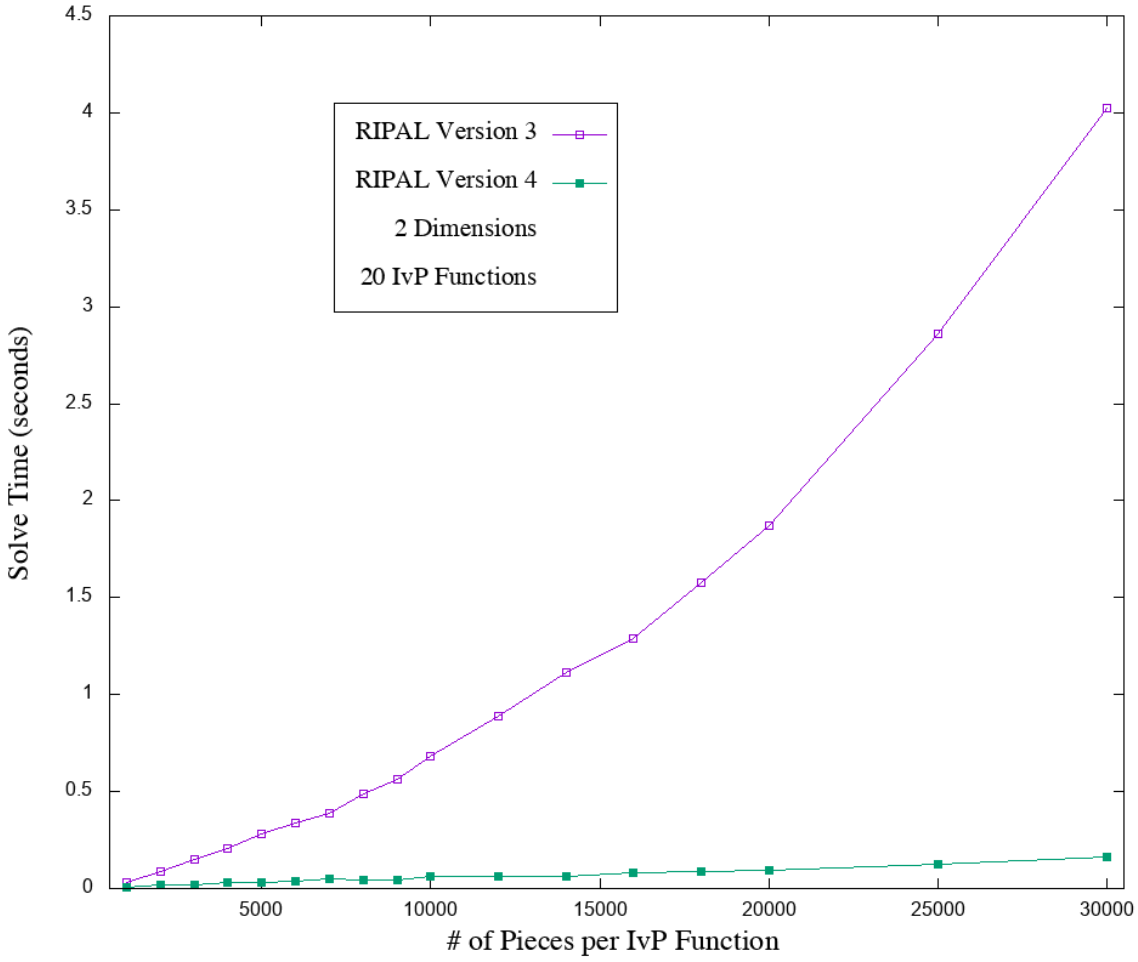

Figure 16: Trial 4A RIPAL: Version 3 vs. Version 4 (2 Dimensions).

6.1.1 Data From Trial 4A - RIPAL Version 3

Table 13 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 16. Time is shown in seconds.

	dim=2, ipfs=20							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.030	0.003	0.02	0.04	5963.92	73.63	5835	6166
2000	0.088	0.004	0.08	0.09	17999.16	244.93	17548	18646
3000	0.152	0.010	0.14	0.18	28353.68	491.55	27618	29344
4000	0.208	0.006	0.20	0.22	37258.88	541.90	36368	38262
5000	0.280	0.015	0.26	0.33	45136.36	906.30	43136	46877
6000	0.337	0.015	0.32	0.36	52089.12	1083.82	49314	53840
7000	0.390	0.009	0.37	0.41	58566.52	1641.34	55644	61162
8000	0.488	0.017	0.46	0.53	64654.88	1281.03	62420	67353
9000	0.564	0.019	0.53	0.60	69502.16	1616.03	66111	71413
10000	0.679	0.025	0.60	0.73	74454.56	1469.96	71272	78465
12000	0.890	0.025	0.84	0.94	83437.68	1838.75	79392	85780
14000	1.112	0.056	0.99	1.21	90103.64	2088.42	86087	94221
16000	1.288	0.044	1.21	1.37	98108.04	2428.51	93331	102043
18000	1.576	0.042	1.51	1.65	105886.36	2938.12	100450	111781
20000	1.871	0.068	1.73	1.99	112480.56	4102.83	105850	119771
25000	2.860	0.069	2.72	3.00	128325.44	3996.77	121039	137847
30000	4.025	0.150	3.73	4.29	145541.88	5445.18	133084	157581

Table 13: Trial 4A `RIPAL` Version 3. All problems have 2 dimensions, 20 IvP functions. Solution time is explored as the number of pieces vary.

6.1.2 Data From Trial 4A - RIPAL Version 4

Table 14 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 16. Time is shown in seconds.

	dim=2, ipfs=20							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.007	0.005	0.00	0.02	30.36	12.15	4	61
2000	0.016	0.011	0.00	0.05	35.36	15.63	12	66
3000	0.022	0.010	0.00	0.04	38.20	13.30	2	58
4000	0.032	0.014	0.01	0.05	46.76	17.88	20	83
5000	0.030	0.014	0.01	0.06	40.84	17.34	10	81
6000	0.037	0.016	0.01	0.08	45.80	13.74	22	83
7000	0.052	0.018	0.02	0.09	53.08	18.06	9	99
8000	0.041	0.016	0.01	0.08	36.48	14.52	5	67
9000	0.045	0.021	0.01	0.11	49.24	18.73	20	108
10000	0.063	0.034	0.02	0.15	50.32	22.82	8	108
12000	0.062	0.032	0.02	0.17	46.44	21.34	12	89
14000	0.064	0.024	0.03	0.15	50.28	18.20	15	79
16000	0.079	0.035	0.02	0.15	46.48	14.21	24	85
18000	0.089	0.040	0.04	0.25	51.44	21.29	18	115
20000	0.095	0.039	0.04	0.17	42.68	15.50	14	76
25000	0.125	0.065	0.05	0.31	40.68	17.50	15	88
30000	0.165	0.074	0.06	0.37	48.00	25.14	17	110

Table 14: Trial 4A `RIPAL` Version 4: All problems have 2 dimensions, 20 IvP functions. Solution time is explored as the number of pieces vary.

6.2 Trial 4B: RIPAL Version 3 vs. Version 4 (4 Dimensions)

In Trial 4B, each IvP problem has four dimensions and twenty IvP functions. The number of pieces is varied between 1000 and 30000 pieces. For each problem type, 25 random instances were generated.

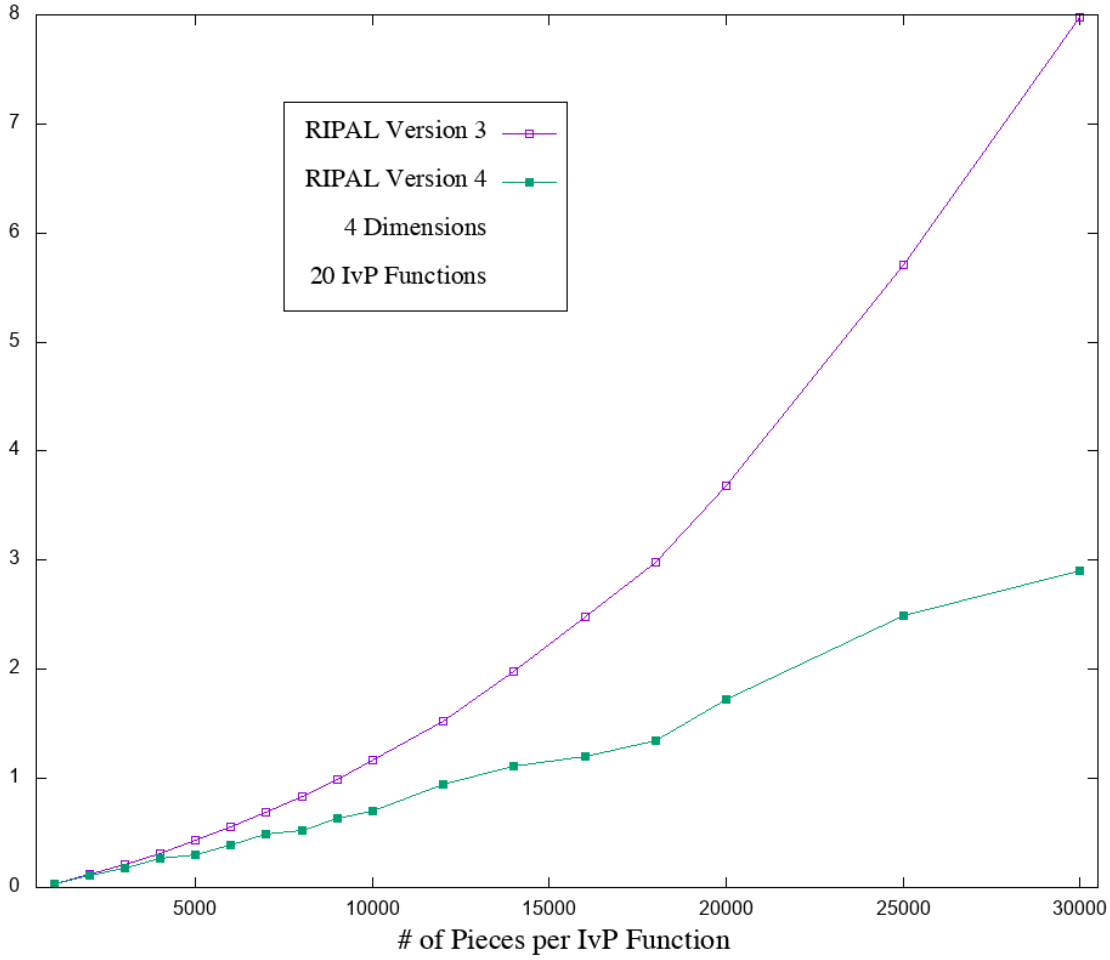

Figure 17: Trial 4B RIPAL: Version 3 vs. Version 4 (4 Dimensions).

6.2.1 Data From Trial 4B - RIPAL Version 3

Table 15 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 17. Time is shown in seconds.

	dim=4, ipfs=20							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.038	0.004	0.03	0.04	5709.28	262.87	5089	6075
2000	0.124	0.007	0.11	0.13	18866.28	997.42	16735	20756
3000	0.215	0.007	0.20	0.23	32517.36	1401.06	28979	35332
4000	0.316	0.008	0.30	0.33	45925.60	1790.27	41855	48758
5000	0.430	0.015	0.39	0.45	59540.00	1659.00	56796	64558
6000	0.554	0.017	0.52	0.60	74089.80	2848.14	68115	80541
7000	0.694	0.022	0.65	0.74	87996.84	4384.31	80690	97988
8000	0.831	0.028	0.78	0.89	103205.24	4494.41	95636	112486
9000	0.994	0.029	0.93	1.06	117911.16	3937.79	111081	127027
10000	1.165	0.065	1.09	1.43	133314.28	5063.25	125207	145846
12000	1.526	0.077	1.45	1.80	162840.64	4651.60	154041	175658
14000	1.977	0.103	1.77	2.19	191335.72	6803.38	176784	206063
16000	2.483	0.145	2.31	2.91	221388.56	8947.73	201918	235489
18000	2.987	0.182	2.63	3.41	253386.16	9320.56	231110	270581
20000	3.684	0.275	3.09	4.20	279395.56	9017.92	257890	300812
25000	5.702	0.518	5.02	6.97	352282.08	14411.93	332059	382077
30000	7.983	0.643	6.86	9.55	438751.16	16257.46	407529	467173

Table 15: Trial 4B `RIPAL` Version 3: All problems have 4 dimensions, 20 IvP functions. Solution time is explored as the number of pieces vary.

6.2.2 Data From Trial 4B - RIPAL Version 4

Table 16 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 17. Time is shown in seconds.

	dim=4, ipfs=20							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
1000	0.038	0.016	0.00	0.07	20.48	9.63	4	39
2000	0.112	0.044	0.03	0.21	21.80	9.30	4	50
3000	0.174	0.063	0.06	0.33	29.16	13.30	4	65
4000	0.270	0.091	0.13	0.47	29.56	13.19	8	65
5000	0.304	0.103	0.10	0.62	26.56	11.65	4	53
6000	0.395	0.153	0.05	0.71	31.96	14.51	10	66
7000	0.489	0.183	0.05	0.84	36.92	14.17	9	74
8000	0.519	0.207	0.15	1.00	29.36	9.28	14	57
9000	0.633	0.226	0.25	1.09	35.24	11.83	18	68
10000	0.702	0.240	0.34	1.32	31.72	19.34	9	84
12000	0.943	0.334	0.38	1.73	34.64	14.25	11	61
14000	1.113	0.447	0.24	1.95	34.80	11.64	9	57
16000	1.198	0.433	0.49	2.03	36.68	9.71	10	54
18000	1.348	0.533	0.47	2.34	38.52	15.88	9	76
20000	1.727	0.719	0.47	3.34	40.28	17.60	10	83
25000	2.489	1.042	0.90	5.09	43.60	16.07	20	79
30000	2.904	1.324	0.51	6.11	44.00	16.03	14	78

Table 16: Trial 4B `RIPAL` Version 4: All problems have 4 dimensions, 20 IvP functions. Solution time is explored as the number of pieces vary.

7 Trial 5: RIPAL Version 3 vs. Version 4 (Varying IvP Functions)

Trial 5 contains a pair of trials, 5A and 5B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 3 and Version 4 are compared. The details of these algorithms are outside the scope here. This trial is similar to Trials 3 and 4 with the primary difference being that the number of pieces per objective function is held constant and the number of IvP functions is varied.

7.1 Trial 5A: RIPAL Version 3 vs. Version 4 (2 Dimensions)

In Trial 5A, each IvP problem has two dimensions and 6000 pieces in each IvP function. The number of IvP functions is varied between 2 and 80 IvP functions. For each problem type, 25 random instances were generated.

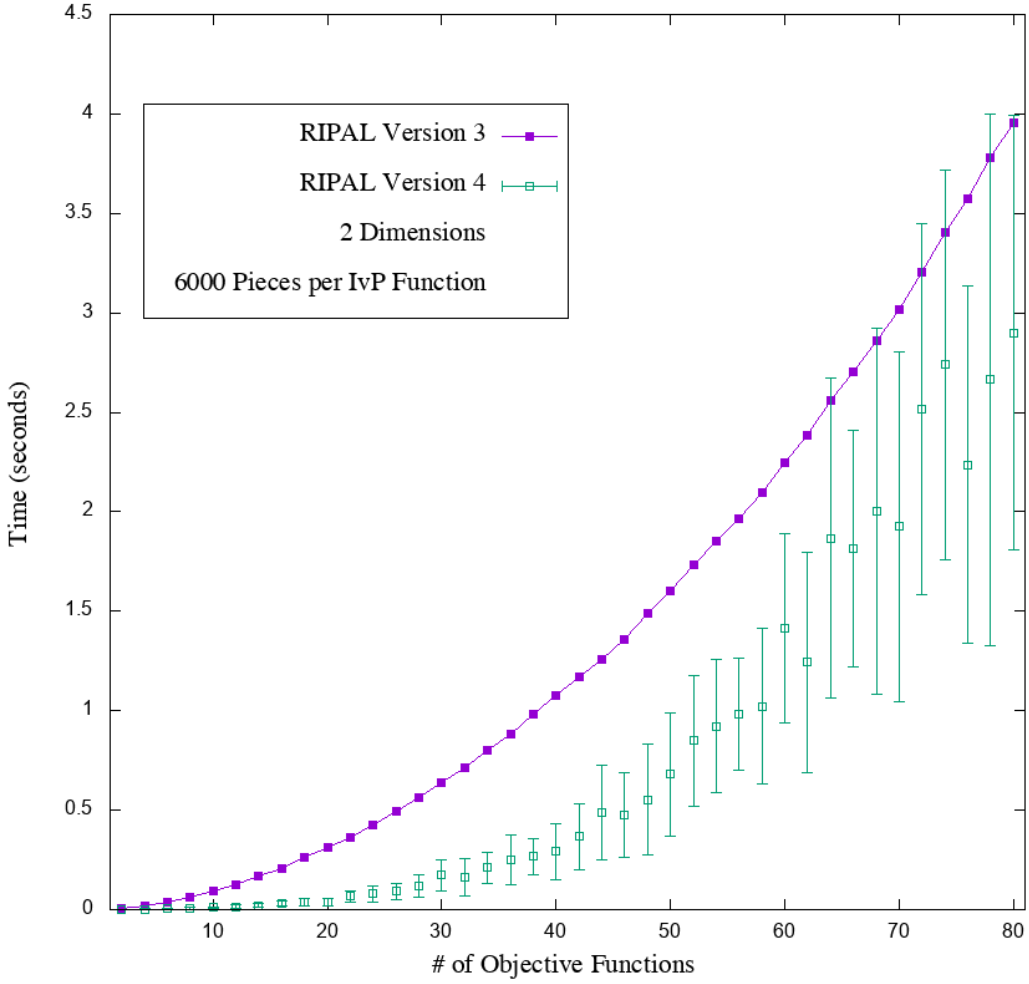

Figure 18: Trial 5A RIPAL: Version 3 vs. Version 4 (2 Dimensions).

7.1.1 Data From Trial 5A - RIPAL Version 3

Table 17 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 18. Time is shown in seconds.

	dim=2, pcs=6000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.003	0.005	0.00	0.01	9141.24	168.63	8774	9500
4	0.018	0.004	0.01	0.02	14622.52	474.71	13498	15439
6	0.038	0.004	0.03	0.04	19783.16	636.45	18254	20561
8	0.063	0.005	0.06	0.07	24054.92	534.31	22614	24833
10	0.092	0.004	0.09	0.10	28206.56	681.70	27019	29729
12	0.127	0.006	0.12	0.14	32282.48	655.68	31135	33556
14	0.167	0.006	0.16	0.18	35982.72	849.70	33991	37777
16	0.210	0.008	0.20	0.23	39621.24	1007.50	37188	41687
18	0.261	0.006	0.25	0.27	43600.64	971.36	41687	45144
20	0.310	0.007	0.30	0.32	46792.88	1022.08	43581	48288
22	0.364	0.010	0.34	0.38	50355.48	831.16	48336	51674
24	0.428	0.008	0.41	0.44	53439.84	873.04	52293	55382
26	0.494	0.013	0.47	0.51	56976.08	1172.30	55118	59447
28	0.564	0.013	0.53	0.59	59785.88	1196.01	57599	61790
30	0.638	0.017	0.59	0.66	62763.68	944.38	60799	64367
32	0.714	0.015	0.69	0.75	66125.32	1526.63	63511	68710
34	0.802	0.014	0.76	0.83	68809.24	1432.29	66458	73433
36	0.881	0.020	0.82	0.91	71682.80	1003.85	69828	73214
38	0.980	0.022	0.93	1.02	74435.32	1703.44	70817	78599
40	1.077	0.024	1.04	1.13	77450.80	1254.33	75072	80283
42	1.168	0.025	1.12	1.22	80262.84	1133.54	77440	82357
44	1.261	0.029	1.21	1.31	82785.24	1090.66	80320	85131
46	1.360	0.029	1.31	1.42	85325.32	1167.00	83063	89053
48	1.491	0.031	1.43	1.54	88550.16	1346.45	85332	90521
50	1.603	0.036	1.54	1.68	90509.12	1509.52	87396	93081
52	1.733	0.040	1.66	1.80	93542.00	1252.37	91400	96394
54	1.854	0.045	1.79	1.94	96160.76	1286.35	92971	98831
56	1.966	0.037	1.89	2.04	97924.60	1347.80	95449	100180
58	2.098	0.044	2.02	2.19	100555.44	1402.43	97994	103183
60	2.246	0.036	2.19	2.32	103673.32	1479.76	100566	106564
62	2.386	0.052	2.26	2.50	105443.04	1662.59	100835	107617
64	2.559	0.038	2.47	2.61	108012.64	1505.49	106230	111198
66	2.705	0.046	2.63	2.82	109960.16	1828.15	106322	113153
68	2.861	0.048	2.78	2.96	112795.88	1324.31	110818	115923
70	3.017	0.041	2.93	3.09	114269.52	1137.67	111993	116703
72	3.205	0.039	3.14	3.30	116722.60	1844.46	113930	120057
74	3.403	0.074	3.29	3.58	118999.36	1585.03	115480	121828
76	3.576	0.055	3.49	3.68	121347.48	1356.80	118913	123923
78	3.782	0.082	3.64	3.95	123646.88	1693.62	120089	126140
80	3.957	0.089	3.77	4.19	125705.96	1437.36	123270	128113

Table 17: Trial 5A `RIPAL` Version 3: All problems have 2 dimensions, 6000 pieces in each IvP function. Solution time is explored as the number of IvP functions vary.

7.1.2 Data From Trial 5A - RIPAL Version 4

Table 18 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 18. Time is shown in seconds.

	dim=2, pcs=6000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.000	0.000	0.00	0.00	48.20	30.63	1	114
4	0.002	0.004	0.00	0.01	54.44	33.17	7	131
6	0.003	0.005	0.00	0.01	55.48	28.09	1	108
8	0.008	0.005	0.00	0.02	64.56	32.73	5	133
10	0.010	0.004	0.00	0.02	67.20	34.30	8	155
12	0.015	0.007	0.00	0.03	53.04	20.50	17	88
14	0.021	0.007	0.01	0.03	69.28	23.88	2	117
16	0.029	0.013	0.01	0.06	60.52	23.48	14	100
18	0.037	0.020	0.01	0.09	53.64	29.88	6	115
20	0.040	0.019	0.01	0.09	55.24	24.16	16	116
22	0.068	0.028	0.02	0.13	63.16	29.21	15	136
24	0.081	0.041	0.02	0.20	56.92	28.88	7	113
26	0.093	0.041	0.03	0.18	53.84	22.86	16	103
28	0.118	0.058	0.02	0.23	63.52	34.45	2	153
30	0.172	0.076	0.03	0.33	64.88	26.58	21	121
32	0.164	0.094	0.05	0.39	66.52	28.90	18	129
34	0.210	0.076	0.09	0.36	73.64	26.70	26	121
36	0.250	0.127	0.04	0.56	65.36	34.12	23	157
38	0.268	0.092	0.10	0.47	64.48	28.51	7	122
40	0.291	0.140	0.06	0.72	60.60	23.03	8	107
42	0.367	0.167	0.06	0.74	64.28	29.04	10	120
44	0.488	0.240	0.13	0.98	71.56	32.94	13	143
46	0.476	0.213	0.12	0.98	65.00	30.05	17	141
48	0.552	0.278	0.16	1.08	65.16	28.30	4	115
50	0.682	0.310	0.13	1.27	70.60	26.23	38	126
52	0.849	0.329	0.14	1.50	74.00	33.98	18	140
54	0.922	0.333	0.09	1.53	75.76	30.01	7	147
56	0.983	0.283	0.54	1.81	67.64	22.46	16	113
58	1.023	0.391	0.29	1.82	78.04	29.74	17	132
60	1.415	0.474	0.17	2.71	70.88	25.02	20	123
62	1.243	0.554	0.40	2.86	65.24	27.89	20	114
64	1.868	0.806	0.63	3.70	80.24	28.19	46	145
66	1.816	0.595	0.93	2.95	76.96	37.03	21	180
68	2.002	0.919	0.25	3.68	81.52	39.14	19	169
70	1.926	0.878	0.69	3.99	59.08	27.89	15	129
72	2.517	0.932	0.54	4.27	80.20	33.09	8	135
74	2.738	0.979	1.00	4.55	83.88	31.32	28	165
76	2.236	0.896	0.42	4.26	64.60	26.20	19	123
78	2.665	1.336	0.55	5.52	77.28	29.98	24	138
80	2.901	1.095	1.48	5.53	72.52	34.81	17	137

Table 18: Trial 5A `RIPAL` Version 4: All problems have 2 dimensions, 6000 pieces in each IvP function. Solution time is explored as the number of IvP functions vary.

7.2 Trial 5B: RIPAL Version 3 vs. Version 4 (4 Dimensions)

In Trial 5B, each IvP problem has four dimensions and 8000 pieces in each IvP function. The number of IvP functions is varied between 2 and 50 IvP functions. For each problem type, 25 random instances were generated.

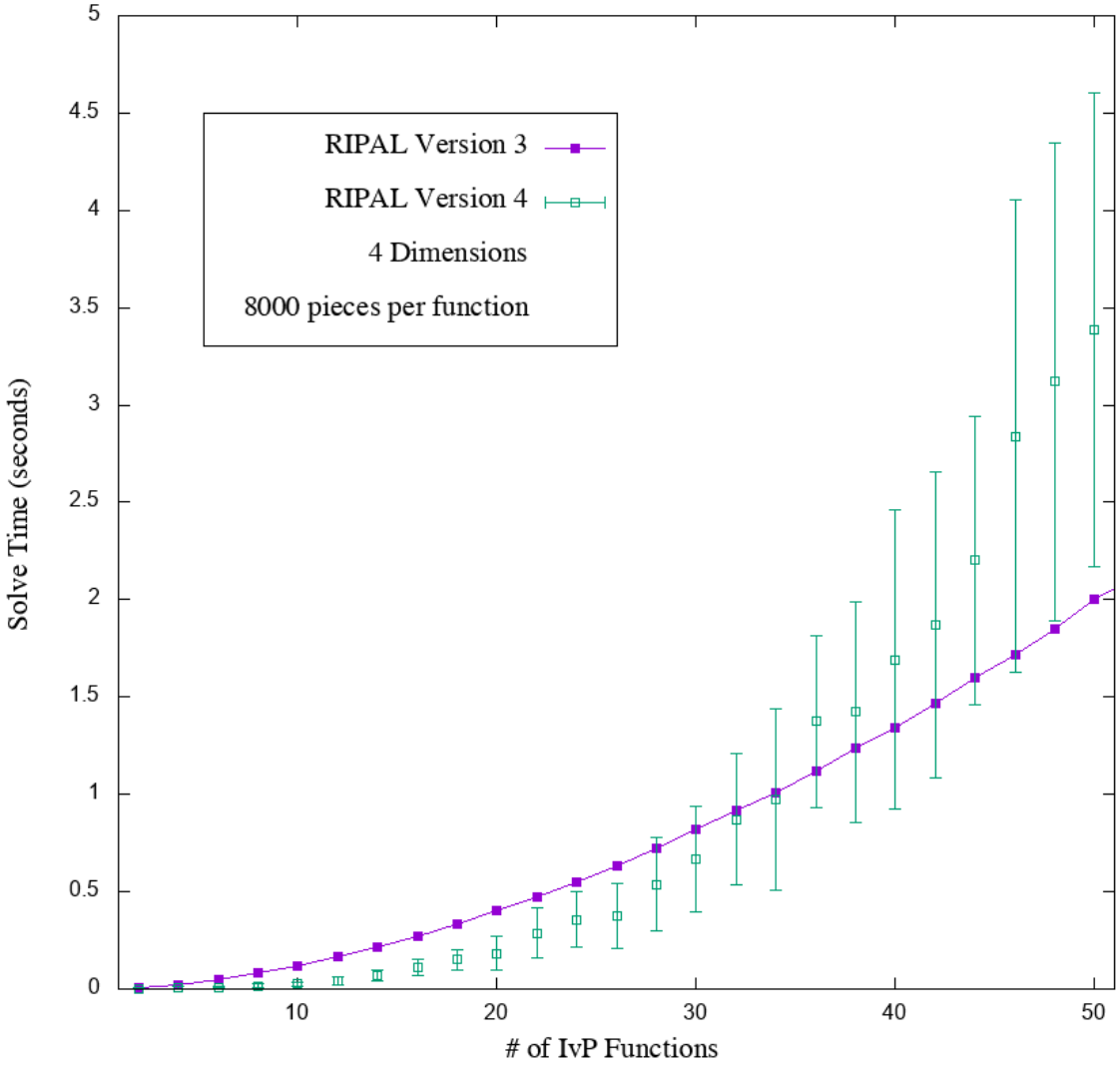

Figure 19: Trial 5B RIPAL: Version 3 vs. Version 4 (4 Dimensions).

7.2.1 Data From Trial 5B - RIPAL Version 3

Table 19 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 19. Time is shown in seconds.

	dim=4, pcs=8000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.005	0.005	0.00	0.01	11752.16	194.81	11311	11904
4	0.024	0.005	0.02	0.03	18794.48	346.87	17919	19290
6	0.050	0.003	0.04	0.06	25611.56	505.35	24761	26621
8	0.084	0.005	0.08	0.09	31902.84	896.67	29783	33540
10	0.122	0.004	0.12	0.13	38023.48	723.49	36099	39158
12	0.168	0.004	0.16	0.17	43918.00	1064.85	40875	45994
14	0.219	0.007	0.20	0.23	49133.64	1220.01	46638	51370
16	0.274	0.008	0.26	0.29	54450.92	1711.99	51268	57116
18	0.336	0.008	0.31	0.35	59386.36	1767.63	54825	62659
20	0.404	0.010	0.39	0.42	65032.00	2320.21	60582	69502
22	0.476	0.011	0.46	0.50	70042.52	2283.44	64561	74018
24	0.551	0.016	0.52	0.58	74428.80	2891.57	67626	79452
26	0.636	0.016	0.59	0.66	78502.72	1979.19	73861	82669
28	0.724	0.022	0.68	0.76	82827.88	3123.51	76899	87766
30	0.817	0.017	0.78	0.86	86856.84	2069.58	82663	90288
32	0.919	0.018	0.88	0.95	91353.40	2630.55	85371	95612
34	1.011	0.028	0.97	1.06	94938.44	3060.75	88282	102151
36	1.123	0.029	1.07	1.19	100004.32	2764.49	94707	105113
38	1.240	0.033	1.18	1.30	103977.88	3086.72	98824	110148
40	1.342	0.037	1.28	1.42	107519.80	3637.69	99957	114997
42	1.464	0.027	1.39	1.52	112212.60	3192.65	106075	118548
44	1.597	0.040	1.52	1.67	115233.64	3221.49	109501	121623
46	1.719	0.045	1.62	1.80	118535.76	3537.88	111951	125069
48	1.849	0.055	1.74	1.95	120861.12	3976.51	110293	127368
50	2.000	0.062	1.84	2.13	127216.48	4114.49	120372	135073
52	2.119	0.055	2.00	2.20	129923.04	3841.80	120731	135491
54	2.299	0.065	2.11	2.45	133824.40	4051.75	123282	143042
56	2.417	0.058	2.30	2.54	136171.28	3623.44	129742	144099
58	2.560	0.061	2.40	2.66	139171.28	3944.68	129930	147636
60	2.740	0.067	2.59	2.88	144294.72	4178.43	136478	151523
62	2.846	0.054	2.74	2.95	144540.32	3046.71	135911	150749
64	3.033	0.067	2.90	3.20	148704.64	3872.89	140740	156719
66	3.232	0.092	3.07	3.39	152347.56	4599.68	144036	162516
68	3.457	0.082	3.31	3.60	157085.08	4043.41	149238	165330
70	3.560	0.081	3.38	3.68	156302.24	3707.40	149384	165515
72	3.756	0.097	3.60	3.97	160454.84	4547.77	152216	170379
74	3.977	0.089	3.78	4.22	164518.44	3546.08	157188	171084
76	4.212	0.121	3.95	4.39	168184.64	4011.74	157654	175070
78	4.437	0.098	4.24	4.61	171701.16	4370.90	162034	179837
80	4.616	0.101	4.39	4.83	173742.56	5627.97	159610	186329

Table 19: Trial 5B `RIPAL` Version 3: All problems have 4 dimensions, 8000 pieces in each IvP function. Solution time is explored as the number of IvP functions vary.

7.2.2 Data From Trial 5B - RIPAL Version 4

Table 20 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 19. Time is shown in seconds.

	dim=4, pcs=8000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.001	0.003	0.00	0.01	30.60	17.30	7	77
4	0.006	0.005	0.00	0.01	36.88	17.58	1	74
6	0.010	0.004	0.00	0.02	32.76	11.98	8	67
8	0.016	0.008	0.00	0.04	38.92	15.42	15	66
10	0.025	0.011	0.00	0.05	33.76	13.96	13	59
12	0.042	0.018	0.02	0.08	33.64	17.42	6	82
14	0.067	0.028	0.02	0.11	33.00	17.65	7	73
16	0.111	0.042	0.04	0.21	40.64	19.48	1	96
18	0.151	0.052	0.06	0.25	41.28	15.67	14	80
20	0.184	0.085	0.03	0.32	42.68	17.59	11	101
22	0.287	0.130	0.04	0.59	41.44	19.51	8	86
24	0.357	0.143	0.11	0.70	42.24	19.02	12	80
26	0.375	0.166	0.09	0.73	34.40	15.53	8	75
28	0.538	0.240	0.18	1.08	40.32	17.08	7	70
30	0.667	0.273	0.25	1.32	40.44	16.09	19	85
32	0.871	0.339	0.06	1.54	36.68	17.23	12	67
34	0.974	0.467	0.20	1.93	36.24	18.12	4	69
36	1.375	0.441	0.52	2.14	40.80	18.08	5	89
38	1.424	0.567	0.26	2.46	36.84	13.51	12	74
40	1.690	0.769	0.14	3.16	35.72	13.67	11	69
42	1.871	0.785	0.54	3.29	40.20	17.37	12	79
44	2.201	0.741	0.77	3.73	39.68	15.02	10	73
46	2.839	1.212	0.51	4.94	42.36	16.39	14	74
48	3.120	1.225	0.90	5.51	39.92	14.78	17	70
50	3.384	1.217	1.22	5.86	40.16	14.99	14	76
52	4.036	1.276	2.09	6.50	38.84	18.27	13	81
54	4.650	1.588	1.57	7.63	40.24	15.23	13	72
56	6.165	1.852	1.07	8.62	39.96	14.85	10	64
58	6.282	1.643	3.33	9.29	38.40	16.88	12	86
60	7.128	1.989	2.68	10.30	41.88	18.01	10	72
62	7.181	2.160	2.12	10.16	38.32	15.91	11	73
64	9.842	2.120	5.35	13.48	41.80	15.87	20	72
66	9.766	2.639	5.18	14.28	39.44	16.71	14	86
68	9.865	3.022	4.02	14.81	41.32	17.59	8	72
70	11.088	3.230	4.02	17.76	40.04	12.82	14	59
72	11.482	2.803	6.63	17.32	42.16	18.09	12	80
74	12.010	3.211	5.79	18.65	40.04	14.71	16	75
76	14.672	3.574	5.93	21.94	45.68	19.53	20	92
78	15.472	4.939	3.68	25.18	43.12	15.64	22	85
80	15.824	3.955	6.83	23.21	40.76	18.48	7	79

Table 20: Trial 5B `RIPAL` Version 4: All problems have 4 dimensions, 8000 pieces in each IvP function. Solution time is explored as the number of IvP functions vary.

8 Trial 6: RIPAL Version 4 vs. Version 5

Trial 6 contains a pair of trials, 6A and 6B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 4 and Version 5 are compared. The details of these algorithms are outside the scope here. In short, RIPAL Version 5 differs from Version 4 only in that an initial solution is calculated to seed the the branch and bound algorithm for perhaps more efficient pruning earlier in the search process.

8.1 Trial 6A: RIPAL Version 4 vs. Version 5 (3 Dimensions)

In Trial 6A, each IvP problem has three dimensions and 25 IvP functions. The number of pieces is varied between 2000 and 40000 pieces. For each problem type, 25 random instances were generated.

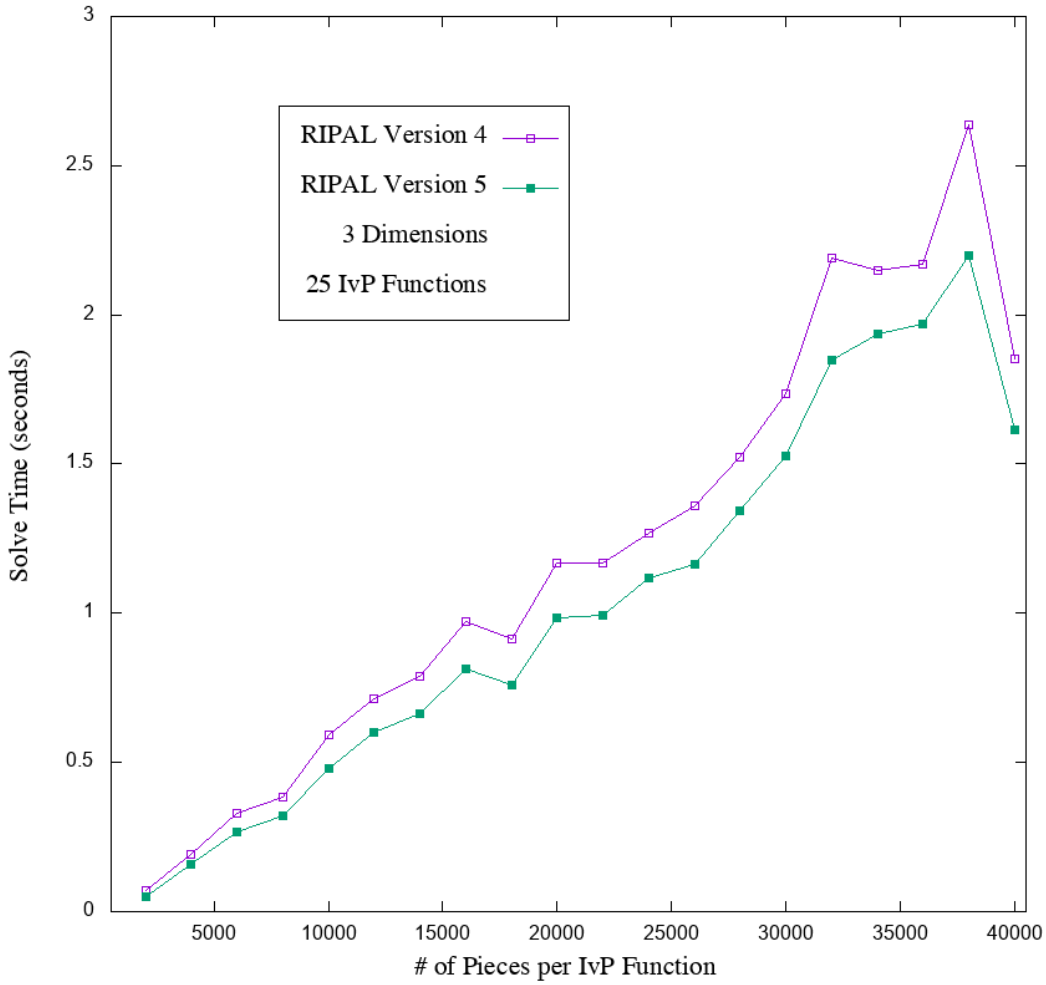

Figure 20: Trial 6A RIPAL: Version 4 vs. Version 5 (3 Dimensions).

8.1.1 Data From Trial 6A - RIPAL Version 4

Table 21 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 20. Time is shown in seconds.

	dim=3, ipfs=25							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
2000	0.072	0.027	0.02	0.13	31.28	13.03	7	61
4000	0.190	0.088	0.06	0.39	35.28	14.69	13	65
6000	0.329	0.172	0.01	0.65	40.20	16.29	8	65
8000	0.386	0.174	0.13	0.72	42.72	15.86	12	75
10000	0.591	0.216	0.15	0.91	46.64	15.27	17	82
12000	0.714	0.279	0.26	1.29	45.72	17.74	16	85
14000	0.790	0.256	0.16	1.22	53.24	18.77	17	87
16000	0.971	0.420	0.10	1.75	54.36	16.68	28	83
18000	0.916	0.459	0.32	2.02	54.92	28.73	7	131
20000	1.167	0.568	0.24	2.75	59.16	26.49	22	115
22000	1.167	0.506	0.28	2.24	57.80	21.58	12	111
24000	1.270	0.628	0.10	2.53	55.72	21.39	24	120
26000	1.362	0.566	0.22	2.27	57.80	19.67	20	99
28000	1.521	0.911	0.16	3.93	54.32	20.23	25	120
30000	1.734	0.626	0.27	2.80	56.40	24.89	21	123
32000	2.190	1.024	0.46	5.01	70.44	30.12	13	136
34000	2.150	0.773	0.81	3.84	56.96	29.03	11	105
36000	2.168	1.144	0.72	4.68	55.00	30.27	15	124
38000	2.636	1.127	0.82	5.05	69.24	23.90	35	120
40000	1.854	1.016	0.40	4.00	56.92	27.79	23	134

Table 21: Trial 6A `RIPAL` Version 4: All problems have 3 dimensions, 25 IvP functions. Solution time is explored as the number of pieces vary.

8.1.2 Data From Trial 6A - RIPAL Version 5

Table 21 contains the full results of RIPAL Version 5, for 25 randomly generated IvP problems, plotted in Figure 20. Time is shown in seconds.

	dim=3, ipfs=25							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
2000	0.052	0.026	0.01	0.10	7.08	5.01	1	17
4000	0.158	0.071	0.06	0.35	11.92	5.68	5	26
6000	0.269	0.150	0.02	0.60	13.56	6.66	5	33
8000	0.322	0.143	0.11	0.68	14.44	7.58	2	31
10000	0.481	0.187	0.11	0.79	16.08	7.71	4	32
12000	0.601	0.274	0.20	1.21	15.24	7.96	4	42
14000	0.662	0.259	0.14	1.09	18.64	8.22	6	36
16000	0.812	0.420	0.09	1.65	18.60	10.55	2	39
18000	0.759	0.409	0.16	1.92	15.24	8.92	1	31
20000	0.984	0.495	0.11	1.99	14.24	7.23	0	33
22000	0.993	0.455	0.27	2.13	16.84	9.16	1	34
24000	1.118	0.612	0.07	2.53	15.16	10.81	0	49
26000	1.166	0.530	0.21	1.85	16.52	8.86	2	38
28000	1.344	0.823	0.13	3.39	18.76	10.49	3	47
30000	1.529	0.572	0.17	2.56	19.44	8.08	7	34
32000	1.849	0.917	0.25	3.67	20.68	9.18	5	38
34000	1.937	0.731	0.71	3.68	18.04	8.94	3	45
36000	1.971	1.112	0.60	4.39	21.40	14.87	0	59
38000	2.197	1.024	0.76	4.66	18.24	10.72	4	40
40000	1.614	0.927	0.40	3.96	21.44	12.07	0	47

Table 22: Trial 6A `RIPAL` Version 5: All problems have 3 dimensions, 25 IvP functions. Solution time is explored as the number of pieces vary.

8.2 Trial 6B: RIPAL Version 4 vs. Version 5 (4 Dimensions)

In Trial 6A, each IvP problem has four dimensions and 25 IvP functions. The number of pieces is varied between 6000 and 40000 pieces. For each problem type, 25 random instances were generated.

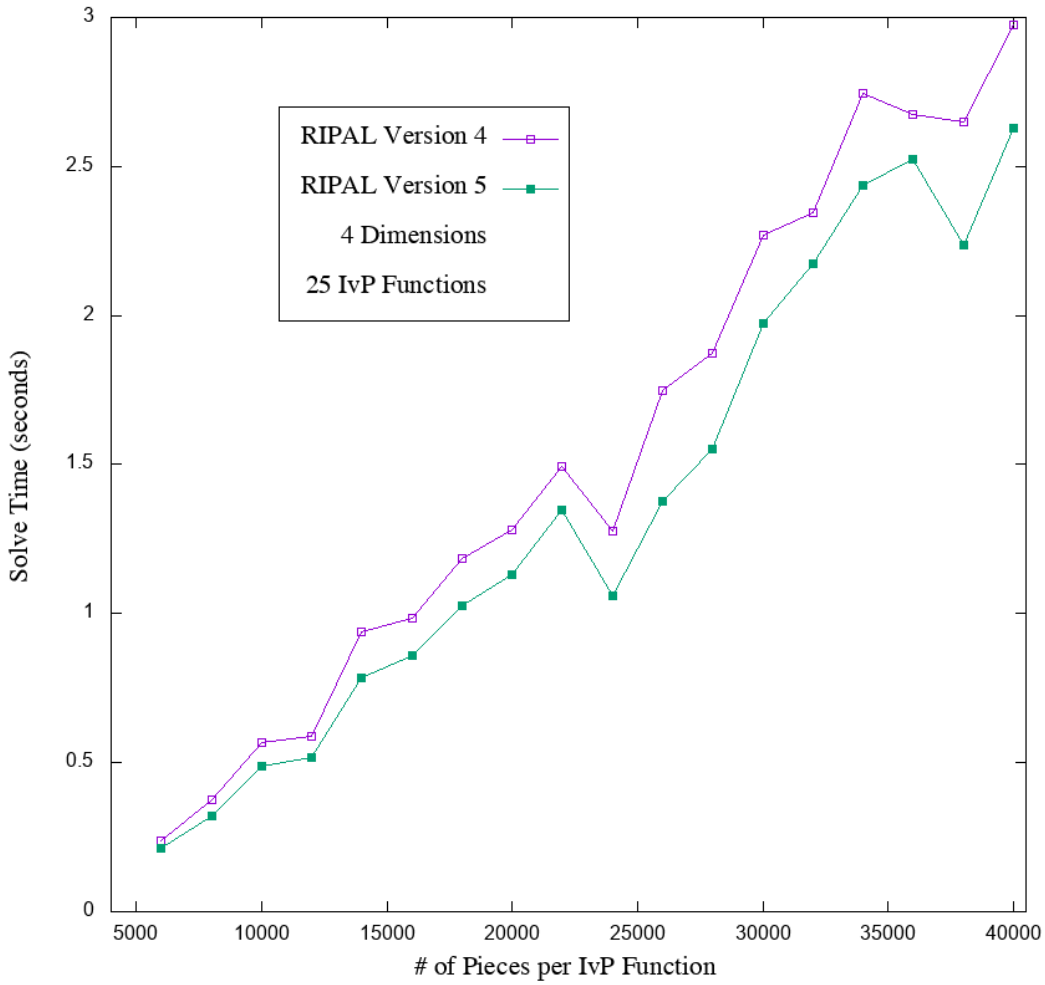

Figure 21: Trial 6B RIPAL: Version 4 vs. Version 5 (4 Dimensions).

8.2.1 Data From Trial 6B - RIPAL Version 4

Table 23 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 21. Time is shown in seconds.

	dim=4, ipfs=25							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
6000	0.236	0.068	0.11	0.38	32.32	12.25	10	52
8000	0.378	0.206	0.04	0.94	34.16	13.25	10	63
10000	0.566	0.229	0.23	1.27	40.56	17.66	8	76
12000	0.588	0.256	0.19	1.02	39.64	17.59	20	92
14000	0.940	0.398	0.33	1.83	48.88	19.31	16	80
16000	0.986	0.405	0.27	1.95	45.84	14.39	19	82
18000	1.185	0.492	0.41	2.00	41.12	20.39	5	82
20000	1.280	0.608	0.40	2.70	48.32	18.81	18	86
22000	1.493	0.645	0.41	2.98	48.80	20.68	8	99
24000	1.276	0.771	0.14	3.55	45.56	19.16	9	85
26000	1.748	0.787	0.42	3.12	53.84	19.57	21	90
28000	1.875	1.090	0.49	3.88	49.72	21.03	18	105
30000	2.269	1.248	0.37	5.33	50.84	16.41	29	85
32000	2.344	1.147	0.39	4.52	48.00	20.19	12	82
34000	2.746	1.061	0.92	5.03	59.36	23.28	24	111
36000	2.674	1.168	0.34	5.31	47.84	20.82	10	95
38000	2.650	1.469	0.43	5.46	55.60	26.59	18	144
40000	2.974	1.084	1.15	6.13	62.56	23.92	12	110

Table 23: Trial 6B `RIPAL`Version 4: All problems have 4 dimensions, 25 IvP functions. Solution time is explored as the number of pieces vary.

8.2.2 Data From Trial 6B - RIPAL Version 5

Table 24 contains the full results of RIPAL Version 5, for 25 randomly generated IvP problems, plotted in Figure 21. Time is shown in seconds.

	dim=4, ipfs=25							
	CPU Time (seconds)				Leaf Nodes			
pcs	avg	sig	min	max	avg	sig	min	max
6000	0.212	0.061	0.10	0.34	8.84	5.20	3	25
8000	0.319	0.197	0.05	0.84	12.20	4.96	1	23
10000	0.489	0.185	0.22	1.08	12.36	5.95	3	27
12000	0.516	0.245	0.11	1.02	13.84	6.08	5	28
14000	0.786	0.371	0.29	1.69	16.56	7.24	5	30
16000	0.858	0.374	0.25	1.86	16.08	8.75	1	37
18000	1.027	0.468	0.17	1.82	14.48	7.76	4	29
20000	1.131	0.609	0.32	2.60	16.76	6.97	5	31
22000	1.347	0.657	0.14	2.94	15.00	7.33	1	31
24000	1.060	0.660	0.14	2.58	12.60	4.68	2	20
26000	1.377	0.685	0.38	3.03	12.76	9.13	1	46
28000	1.552	0.985	0.45	3.73	16.80	7.14	4	40
30000	1.975	1.121	0.37	4.85	13.56	7.73	0	35
32000	2.174	1.092	0.39	4.42	17.32	8.32	0	33
34000	2.437	1.170	0.68	4.97	18.60	8.19	4	35
36000	2.526	1.179	0.33	5.06	18.04	8.47	2	38
38000	2.235	1.290	0.39	5.00	18.08	8.92	1	36
40000	2.629	0.988	0.97	5.37	19.08	8.69	4	37

Table 24: Trial 6B `RIPAL`Version 5: All problems have 4 dimensions, 25 IvP functions. Solution time is explored as the number of pieces vary.

9 Trial 7: RIPAL Version 4 vs. Version 5 (Varying IvP Functions)

Trial 7 contains a pair of trials, 7A and 7B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 4 and Version 5 are compared. The details of these algorithms are outside the scope here. In short, RIPAL Version 5 differs from Version 4 only in that an initial solution is calculated to seed the branch and bound algorithm for perhaps more efficient pruning earlier in the search process. Trial 7 differs from Trial 6 in that the pieces are held constant and the number of IvP functions are varied within each trial.

9.1 Trial 7A: RIPAL Version 4 vs. Version 5 (2 Dimensions)

In Trial 7A, each IvP problem has two dimensions and 6000 pieces per IvP function. The number of IvP functions is varied between 2 and 80 IvP functions. For each problem type, 25 random instances were generated.

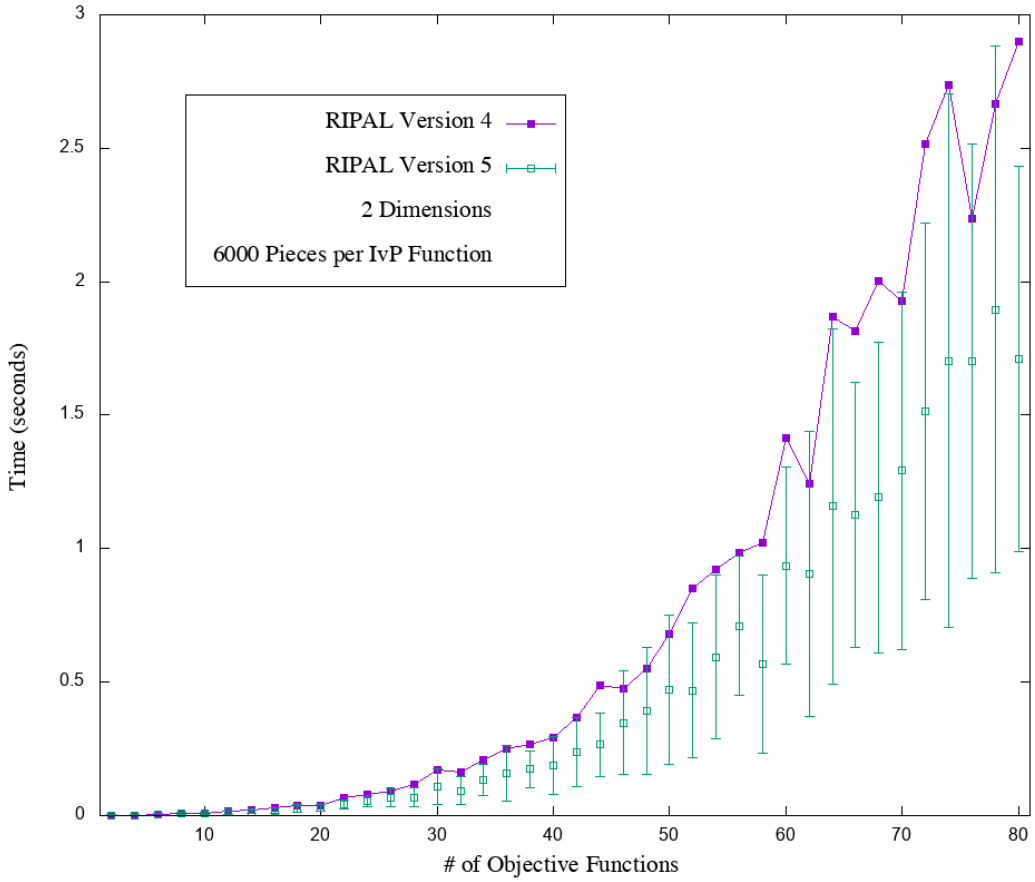

Figure 22: Trial 7A RIPAL: Version 4 vs. Version 5 (2 Dimensions).

9.1.1 Data From Trial 7A - RIPAL Version 4

Table 25 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 22. Time is shown in seconds.

	dim=2, pcs=6000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.000	0.000	0.00	0.00	48.20	30.63	1	114
4	0.002	0.004	0.00	0.01	54.44	33.17	7	131
6	0.003	0.005	0.00	0.01	55.48	28.09	1	108
8	0.008	0.005	0.00	0.02	64.56	32.73	5	133
10	0.010	0.004	0.00	0.02	67.20	34.30	8	155
12	0.015	0.007	0.00	0.03	53.04	20.50	17	88
14	0.021	0.007	0.01	0.03	69.28	23.88	2	117
16	0.029	0.013	0.01	0.06	60.52	23.48	14	100
18	0.037	0.020	0.01	0.09	53.64	29.88	6	115
20	0.040	0.019	0.01	0.09	55.24	24.16	16	116
22	0.068	0.028	0.02	0.13	63.16	29.21	15	136
24	0.081	0.041	0.02	0.20	56.92	28.88	7	113
26	0.093	0.041	0.03	0.18	53.84	22.86	16	103
28	0.118	0.058	0.02	0.23	63.52	34.45	2	153
30	0.172	0.076	0.03	0.33	64.88	26.58	21	121
32	0.164	0.094	0.05	0.39	66.52	28.90	18	129
34	0.210	0.076	0.09	0.36	73.64	26.70	26	121
36	0.250	0.127	0.04	0.56	65.36	34.12	23	157
38	0.268	0.092	0.10	0.47	64.48	28.51	7	122
40	0.291	0.140	0.06	0.72	60.60	23.03	8	107
42	0.367	0.167	0.06	0.74	64.28	29.04	10	120
44	0.488	0.240	0.13	0.98	71.56	32.94	13	143
46	0.476	0.213	0.12	0.98	65.00	30.05	17	141
48	0.552	0.278	0.16	1.08	65.16	28.30	4	115
50	0.682	0.310	0.13	1.27	70.60	26.23	38	126
52	0.849	0.329	0.14	1.50	74.00	33.98	18	140
54	0.922	0.333	0.09	1.53	75.76	30.01	7	147
56	0.983	0.283	0.54	1.81	67.64	22.46	16	113
58	1.023	0.391	0.29	1.82	78.04	29.74	17	132
60	1.415	0.474	0.17	2.71	70.88	25.02	20	123
62	1.243	0.554	0.40	2.86	65.24	27.89	20	114
64	1.868	0.806	0.63	3.70	80.24	28.19	46	145
66	1.816	0.595	0.93	2.95	76.96	37.03	21	180
68	2.002	0.919	0.25	3.68	81.52	39.14	19	169
70	1.926	0.878	0.69	3.99	59.08	27.89	15	129
72	2.517	0.932	0.54	4.27	80.20	33.09	8	135
74	2.738	0.979	1.00	4.55	83.88	31.32	28	165
76	2.236	0.896	0.42	4.26	64.60	26.20	19	123
78	2.665	1.336	0.55	5.52	77.28	29.98	24	138
80	2.901	1.095	1.48	5.53	72.52	34.81	17	137

Table 25: Trial 7A `RIPAL` Version 4: All problems have 2 dimensions, 6000 pieces. Solution time is explored as the number of IvP functions vary.

9.1.2 Data From Trial 7A - RIPAL Version 5

Table 26 contains the full results of RIPAL Version 5, for 25 randomly generated IvP problems, plotted in Figure 22. Time is shown in seconds.

	dim=2, pcs=6000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.001	0.003	0.00	0.01	27.36	25.19	0	109
4	0.001	0.003	0.00	0.01	19.12	19.29	0	94
6	0.003	0.005	0.00	0.01	14.12	11.85	0	58
8	0.009	0.003	0.00	0.01	19.12	18.20	1	85
10	0.008	0.005	0.00	0.02	16.24	10.08	0	35
12	0.011	0.005	0.00	0.02	12.44	8.96	0	35
14	0.013	0.007	0.00	0.03	14.44	10.39	0	39
16	0.018	0.010	0.00	0.04	15.60	9.03	2	34
18	0.026	0.015	0.01	0.07	12.76	9.20	1	39
20	0.028	0.013	0.01	0.06	18.00	9.68	6	38
22	0.042	0.015	0.02	0.08	13.92	10.93	0	36
24	0.055	0.021	0.02	0.10	12.60	8.05	1	36
26	0.067	0.032	0.02	0.16	15.72	9.40	2	36
28	0.068	0.036	0.02	0.14	13.56	7.24	1	28
30	0.108	0.068	0.01	0.28	13.00	9.06	2	33
32	0.093	0.051	0.02	0.21	13.60	10.33	0	37
34	0.135	0.058	0.02	0.27	16.64	11.57	0	44
36	0.158	0.104	0.03	0.48	17.76	13.28	1	49
38	0.173	0.070	0.06	0.32	16.16	7.68	5	30
40	0.188	0.108	0.05	0.42	13.44	9.92	0	43
42	0.239	0.130	0.06	0.57	10.96	7.50	0	26
44	0.265	0.120	0.11	0.55	13.20	6.86	1	30
46	0.348	0.194	0.09	0.88	15.92	9.96	0	43
48	0.392	0.237	0.10	1.04	13.04	8.02	2	34
50	0.472	0.280	0.08	0.99	16.40	10.16	0	39
52	0.469	0.252	0.06	1.07	10.80	7.02	1	33
54	0.594	0.308	0.07	1.23	15.88	8.76	2	42
56	0.711	0.262	0.25	1.25	12.00	7.86	0	33
58	0.567	0.335	0.10	1.37	11.56	10.84	0	41
60	0.936	0.370	0.15	1.59	13.48	9.31	0	37
62	0.906	0.534	0.18	2.55	12.40	9.29	4	50
64	1.158	0.666	0.17	2.71	17.92	11.53	4	45
66	1.127	0.496	0.44	2.32	14.56	9.56	1	35
68	1.192	0.583	0.21	2.50	13.24	5.75	0	25
70	1.292	0.669	0.07	3.34	9.36	4.77	2	25
72	1.515	0.706	0.54	3.57	12.92	8.65	0	32
74	1.704	1.000	0.41	3.93	14.72	9.15	1	44
76	1.701	0.813	0.38	3.73	12.96	6.68	1	29
78	1.895	0.987	0.54	3.70	10.88	6.75	0	31
80	1.710	0.721	0.46	2.98	9.28	5.50	0	24

Table 26: Trial 7A `RIPAL` Version 5: All problems have 2 dimensions, 6000 pieces. Solution time is explored as the number of IvP functions vary.

9.2 Trial 7B: RIPAL Version 4 vs. Version 5 (4 Dimensions)

In Trial 7B, each IvP problem has four dimensions and 8000 pieces per IvP function. The number of IvP functions is varied between 2 and 80 IvP functions. For each problem type, 25 random instances were generated.

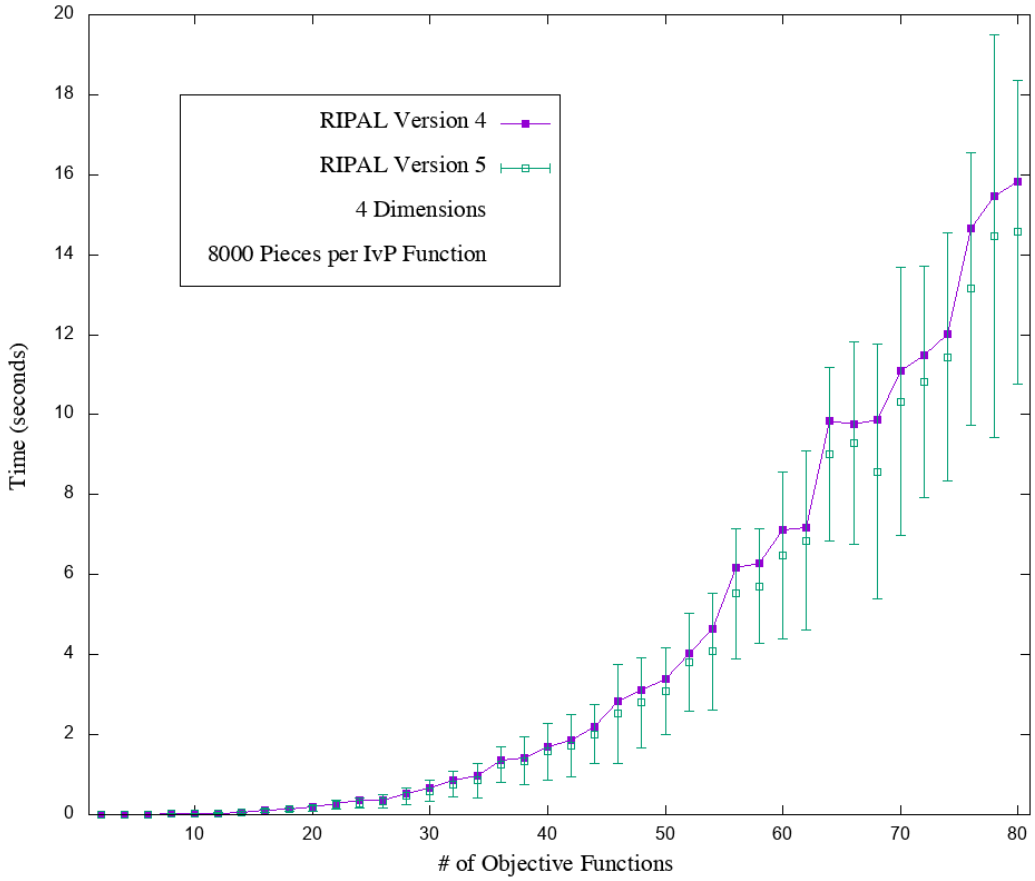

Figure 23: Trial 7B RIPAL: Version 4 vs. Version 5 (4 Dimensions).

9.2.1 Data From Trial 7B - RIPAL Version 4

Table 27 contains the full results of RIPAL Version 4, for 25 randomly generated IvP problems, plotted in Figure 23. Time is shown in seconds.

	dim=4, pcs=8000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.001	0.003	0.00	0.01	30.60	17.30	7	77
4	0.006	0.005	0.00	0.01	36.88	17.58	1	74
6	0.010	0.004	0.00	0.02	32.76	11.98	8	67
8	0.016	0.008	0.00	0.04	38.92	15.42	15	66
10	0.025	0.011	0.00	0.05	33.76	13.96	13	59
12	0.042	0.018	0.02	0.08	33.64	17.42	6	82
14	0.067	0.028	0.02	0.11	33.00	17.65	7	73
16	0.111	0.042	0.04	0.21	40.64	19.48	1	96
18	0.151	0.052	0.06	0.25	41.28	15.67	14	80
20	0.184	0.085	0.03	0.32	42.68	17.59	11	101
22	0.287	0.130	0.04	0.59	41.44	19.51	8	86
24	0.357	0.143	0.11	0.70	42.24	19.02	12	80
26	0.375	0.166	0.09	0.73	34.40	15.53	8	75
28	0.538	0.240	0.18	1.08	40.32	17.08	7	70
30	0.667	0.273	0.25	1.32	40.44	16.09	19	85
32	0.871	0.339	0.06	1.54	36.68	17.23	12	67
34	0.974	0.467	0.20	1.93	36.24	18.12	4	69
36	1.375	0.441	0.52	2.14	40.80	18.08	5	89
38	1.424	0.567	0.26	2.46	36.84	13.51	12	74
40	1.690	0.769	0.14	3.16	35.72	13.67	11	69
42	1.871	0.785	0.54	3.29	40.20	17.37	12	79
44	2.201	0.741	0.77	3.73	39.68	15.02	10	73
46	2.839	1.212	0.51	4.94	42.36	16.39	14	74
48	3.120	1.225	0.90	5.51	39.92	14.78	17	70
50	3.384	1.217	1.22	5.86	40.16	14.99	14	76
52	4.036	1.276	2.09	6.50	38.84	18.27	13	81
54	4.650	1.588	1.57	7.63	40.24	15.23	13	72
56	6.165	1.852	1.07	8.62	39.96	14.85	10	64
58	6.282	1.643	3.33	9.29	38.40	16.88	12	86
60	7.128	1.989	2.68	10.30	41.88	18.01	10	72
62	7.181	2.160	2.12	10.16	38.32	15.91	11	73
64	9.842	2.120	5.35	13.48	41.80	15.87	20	72
66	9.766	2.639	5.18	14.28	39.44	16.71	14	86
68	9.865	3.022	4.02	14.81	41.32	17.59	8	72
70	11.088	3.230	4.02	17.76	40.04	12.82	14	59
72	11.482	2.803	6.63	17.32	42.16	18.09	12	80
74	12.010	3.211	5.79	18.65	40.04	14.71	16	75
76	14.672	3.574	5.93	21.94	45.68	19.53	20	92
78	15.472	4.939	3.68	25.18	43.12	15.64	22	85
80	15.824	3.955	6.83	23.21	40.76	18.48	7	79

Table 27: Trial 7B `RIPAL` Version 4: All problems have 4 dimensions, 8000 pieces. Solution time is explored as the number of IvP functions vary.

9.3 Data From Trial 7B - RIPAL Version 5

Table 28 contains the full results of RIPAL Version 5, for 25 randomly generated IvP problems, plotted in Figure 23. Time is shown in seconds.

	dim=4, pcs=8000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.002	0.004	0.00	0.01	12.96	9.16	1	34
4	0.004	0.005	0.00	0.01	9.44	5.94	1	19
6	0.009	0.003	0.00	0.01	12.80	5.65	3	25
8	0.015	0.008	0.00	0.03	12.68	6.39	3	34
10	0.024	0.011	0.01	0.05	11.72	7.19	0	27
12	0.037	0.017	0.01	0.08	9.96	6.10	1	24
14	0.060	0.024	0.02	0.11	11.68	6.94	2	31
16	0.095	0.041	0.04	0.17	12.12	6.75	0	28
18	0.130	0.047	0.05	0.24	14.48	5.24	2	26
20	0.156	0.073	0.03	0.28	13.28	5.40	5	25
22	0.246	0.118	0.05	0.47	11.20	6.33	0	27
24	0.292	0.126	0.07	0.65	10.68	5.55	0	22
26	0.329	0.170	0.09	0.72	10.56	7.13	0	31
28	0.466	0.206	0.15	0.94	11.84	4.32	5	25
30	0.592	0.268	0.24	1.29	13.96	6.51	2	28
32	0.764	0.316	0.06	1.52	10.64	5.28	1	21
34	0.862	0.431	0.14	1.92	9.72	5.50	1	22
36	1.241	0.445	0.50	2.08	12.56	6.73	0	31
38	1.346	0.593	0.24	2.45	12.28	6.66	1	24
40	1.576	0.704	0.11	2.90	10.52	5.26	0	24
42	1.728	0.772	0.32	3.04	11.40	4.78	3	22
44	2.016	0.731	0.66	3.51	11.40	5.58	3	23
46	2.522	1.236	0.49	4.76	12.72	3.61	6	21
48	2.797	1.138	0.58	5.46	11.96	6.38	3	34
50	3.086	1.079	1.22	5.33	10.16	4.93	3	22
52	3.801	1.222	1.83	6.03	12.16	5.77	1	29
54	4.078	1.453	1.54	7.48	10.84	4.36	1	18
56	5.526	1.632	1.07	8.26	10.84	4.93	5	27
58	5.707	1.428	2.86	8.84	10.40	6.25	0	23
60	6.490	2.083	2.75	10.18	12.44	7.46	2	31
62	6.853	2.233	2.18	10.34	13.48	6.95	2	30
64	9.010	2.169	5.21	12.20	11.40	5.61	1	23
66	9.281	2.531	5.07	13.63	10.44	4.48	0	20
68	8.576	3.179	3.19	13.55	10.84	7.49	0	32
70	10.330	3.349	4.03	17.76	11.24	4.96	4	26
72	10.821	2.883	5.82	16.88	13.12	7.21	4	33
74	11.439	3.103	5.68	18.48	13.60	6.69	4	28
76	13.146	3.409	5.84	17.75	11.64	6.75	0	27
78	14.459	5.037	3.43	25.07	12.48	6.84	1	25
80	14.563	3.805	6.59	21.81	11.72	6.97	0	29

Table 28: Trial 7B `RIPAL`Version 5: All problems have 4 dimensions, 8000 pieces. Solution time is explored as the number of IvP functions vary.

10 Trial 8: RIPAL Version 3 vs. Version 5 (Varying IvP Functions)

Trial 8 contains a pair of trials, 8A and 8B. In each trial the Recursive Interval Programming Algorithm (RIPAL) Version 3 and Version 5 are compared. The details of these algorithms are outside the scope here. In short, RIPAL Version 5 differs from Version 3 only in that an initial solution is calculated to seed the the branch and bound algorithm and Version 3 performs no upper bound calculations. Trial 8 differs from Trial 7 only in that Version 3 is used here instead of Version 4 in Trial 7.

10.1 Trial 8A: RIPAL Version 3 vs. Version 5 (2 Dimensions)

In Trial 8A, each IvP problem has two dimensions and 6000 pieces per IvP function. The number of IvP functions is varied between 2 and 80 IvP functions. For each problem type, 25 random instances were generated.

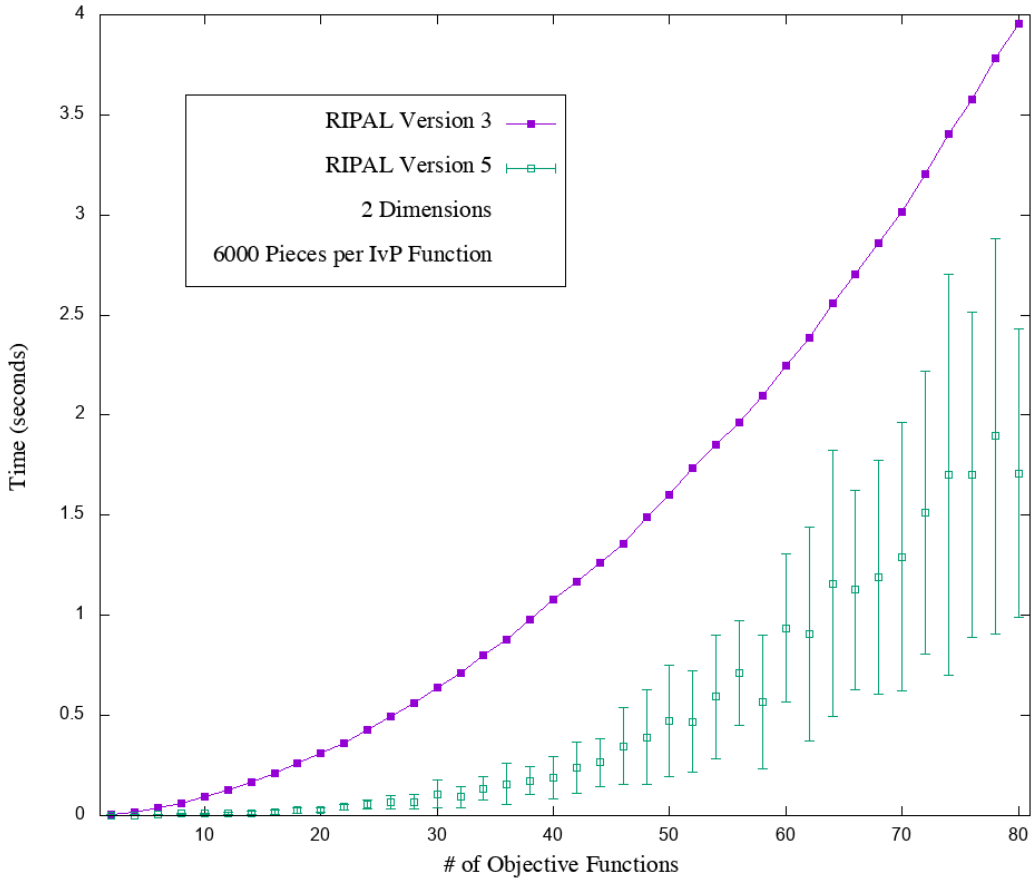

Figure 24: Trial 8A RIPAL: Version 3 vs. Version 5 (2 Dimensions).

10.1.1 Data From Trial 8A - RIPAL Version 3

Table 29 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 24. Time is shown in seconds.

	dim=2, pcs=6000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.003	0.005	0.00	0.01	9141.24	168.63	8774	9500
4	0.018	0.004	0.01	0.02	14622.52	474.71	13498	15439
6	0.038	0.004	0.03	0.04	19783.16	636.45	18254	20561
8	0.063	0.005	0.06	0.07	24054.92	534.31	22614	24833
10	0.092	0.004	0.09	0.10	28206.56	681.70	27019	29729
12	0.127	0.006	0.12	0.14	32282.48	655.68	31135	33556
14	0.167	0.006	0.16	0.18	35982.72	849.70	33991	37777
16	0.210	0.008	0.20	0.23	39621.24	1007.50	37188	41687
18	0.261	0.006	0.25	0.27	43600.64	971.36	41687	45144
20	0.310	0.007	0.30	0.32	46792.88	1022.08	43581	48288
22	0.364	0.010	0.34	0.38	50355.48	831.16	48336	51674
24	0.428	0.008	0.41	0.44	53439.84	873.04	52293	55382
26	0.494	0.013	0.47	0.51	56976.08	1172.30	55118	59447
28	0.564	0.013	0.53	0.59	59785.88	1196.01	57599	61790
30	0.638	0.017	0.59	0.66	62763.68	944.38	60799	64367
32	0.714	0.015	0.69	0.75	66125.32	1526.63	63511	68710
34	0.802	0.014	0.76	0.83	68809.24	1432.29	66458	73433
36	0.881	0.020	0.82	0.91	71682.80	1003.85	69828	73214
38	0.980	0.022	0.93	1.02	74435.32	1703.44	70817	78599
40	1.077	0.024	1.04	1.13	77450.80	1254.33	75072	80283
42	1.168	0.025	1.12	1.22	80262.84	1133.54	77440	82357
44	1.261	0.029	1.21	1.31	82785.24	1090.66	80320	85131
46	1.360	0.029	1.31	1.42	85325.32	1167.00	83063	89053
48	1.491	0.031	1.43	1.54	88550.16	1346.45	85332	90521
50	1.603	0.036	1.54	1.68	90509.12	1509.52	87396	93081
52	1.733	0.040	1.66	1.80	93542.00	1252.37	91400	96394
54	1.854	0.045	1.79	1.94	96160.76	1286.35	92971	98831
56	1.966	0.037	1.89	2.04	97924.60	1347.80	95449	100180
58	2.098	0.044	2.02	2.19	100555.44	1402.43	97994	103183
60	2.246	0.036	2.19	2.32	103673.32	1479.76	100566	106564
62	2.386	0.052	2.26	2.50	105443.04	1662.59	100835	107617
64	2.559	0.038	2.47	2.61	108012.64	1505.49	106230	111198
66	2.705	0.046	2.63	2.82	109960.16	1828.15	106322	113153
68	2.861	0.048	2.78	2.96	112795.88	1324.31	110818	115923
70	3.017	0.041	2.93	3.09	114269.52	1137.67	111993	116703
72	3.205	0.039	3.14	3.30	116722.60	1844.46	113930	120057
74	3.403	0.074	3.29	3.58	118999.36	1585.03	115480	121828
76	3.576	0.055	3.49	3.68	121347.48	1356.80	118913	123923
78	3.782	0.082	3.64	3.95	123646.88	1693.62	120089	126140
80	3.957	0.089	3.77	4.19	125705.96	1437.36	123270	128113

Table 29: Trial 8A `RIPAL` Version 3: All problems have 2 dimensions, 6000 pieces. Solution time is explored as the number of IvP functions vary.

Data From Trial 8A - RIPAL Version 5

Table 30 contains the full results of RIPAL Version 5, for 25 randomly generated IvP problems, plotted in Figure 24. Time is shown in seconds.

	dim=2, pcs=6000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.001	0.003	0.00	0.01	27.36	25.19	0	109
4	0.001	0.003	0.00	0.01	19.12	19.29	0	94
6	0.003	0.005	0.00	0.01	14.12	11.85	0	58
8	0.009	0.003	0.00	0.01	19.12	18.20	1	85
10	0.008	0.005	0.00	0.02	16.24	10.08	0	35
12	0.011	0.005	0.00	0.02	12.44	8.96	0	35
14	0.013	0.007	0.00	0.03	14.44	10.39	0	39
16	0.018	0.010	0.00	0.04	15.60	9.03	2	34
18	0.026	0.015	0.01	0.07	12.76	9.20	1	39
20	0.028	0.013	0.01	0.06	18.00	9.68	6	38
22	0.042	0.015	0.02	0.08	13.92	10.93	0	36
24	0.055	0.021	0.02	0.10	12.60	8.05	1	36
26	0.067	0.032	0.02	0.16	15.72	9.40	2	36
28	0.068	0.036	0.02	0.14	13.56	7.24	1	28
30	0.108	0.068	0.01	0.28	13.00	9.06	2	33
32	0.093	0.051	0.02	0.21	13.60	10.33	0	37
34	0.135	0.058	0.02	0.27	16.64	11.57	0	44
36	0.158	0.104	0.03	0.48	17.76	13.28	1	49
38	0.173	0.070	0.06	0.32	16.16	7.68	5	30
40	0.188	0.108	0.05	0.42	13.44	9.92	0	43
42	0.239	0.130	0.06	0.57	10.96	7.50	0	26
44	0.265	0.120	0.11	0.55	13.20	6.86	1	30
46	0.348	0.194	0.09	0.88	15.92	9.96	0	43
48	0.392	0.237	0.10	1.04	13.04	8.02	2	34
50	0.472	0.280	0.08	0.99	16.40	10.16	0	39
52	0.469	0.252	0.06	1.07	10.80	7.02	1	33
54	0.594	0.308	0.07	1.23	15.88	8.76	2	42
56	0.711	0.262	0.25	1.25	12.00	7.86	0	33
58	0.567	0.335	0.10	1.37	11.56	10.84	0	41
60	0.936	0.370	0.15	1.59	13.48	9.31	0	37
62	0.906	0.534	0.18	2.55	12.40	9.29	4	50
64	1.158	0.666	0.17	2.71	17.92	11.53	4	45
66	1.127	0.496	0.44	2.32	14.56	9.56	1	35
68	1.192	0.583	0.21	2.50	13.24	5.75	0	25
70	1.292	0.669	0.07	3.34	9.36	4.77	2	25
72	1.515	0.706	0.54	3.57	12.92	8.65	0	32
74	1.704	1.000	0.41	3.93	14.72	9.15	1	44
76	1.701	0.813	0.38	3.73	12.96	6.68	1	29
78	1.895	0.987	0.54	3.70	10.88	6.75	0	31
80	1.710	0.721	0.46	2.98	9.28	5.50	0	24

Table 30: Trail 8A `RIPAL` Version 5: All problems have 2 dimensions, 6000 pieces. Solution time is explored as the number of IvP functions vary.

10.2 Trial 8B: RIPAL Version 3 vs. Version 5 (4 Dimensions)

In Trial 8A, each IvP problem has two dimensions and 6000 pieces per IvP function. The number of IvP functions is varied between 2 and 80 IvP functions. For each problem type, 25 random instances were generated.

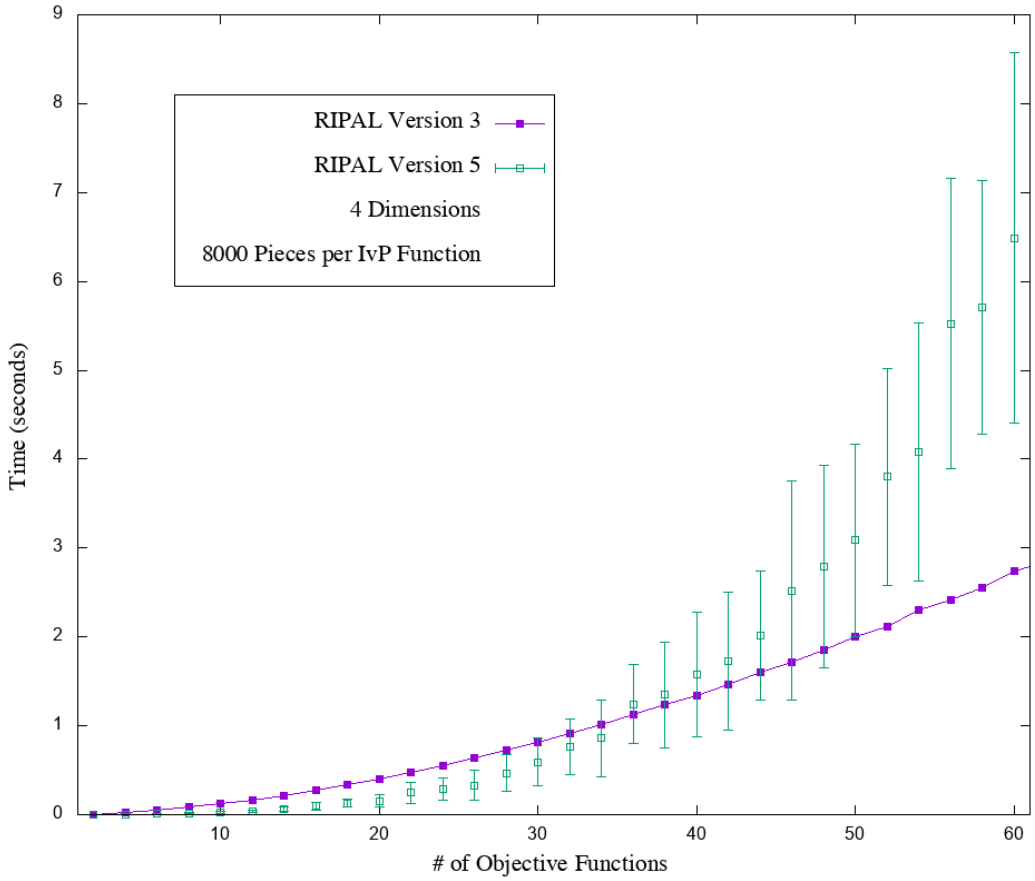

Figure 25: Trial 8B RIPAL: Version 3 vs. Version 5 (4 Dimensions).

10.2.1 Data From Trial 8B - RIPAL Version 3

Table 31 contains the full results of RIPAL Version 3, for 25 randomly generated IvP problems, plotted in Figure 25. Time is shown in seconds.

	dim=4, pcs=8000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.005	0.005	0.00	0.01	11752.16	194.81	11311	11904
4	0.024	0.005	0.02	0.03	18794.48	346.87	17919	19290
6	0.050	0.003	0.04	0.06	25611.56	505.35	24761	26621
8	0.084	0.005	0.08	0.09	31902.84	896.67	29783	33540
10	0.122	0.004	0.12	0.13	38023.48	723.49	36099	39158
12	0.168	0.004	0.16	0.17	43918.00	1064.85	40875	45994
14	0.219	0.007	0.20	0.23	49133.64	1220.01	46638	51370
16	0.274	0.008	0.26	0.29	54450.92	1711.99	51268	57116
18	0.336	0.008	0.31	0.35	59386.36	1767.63	54825	62659
20	0.404	0.010	0.39	0.42	65032.00	2320.21	60582	69502
22	0.476	0.011	0.46	0.50	70042.52	2283.44	64561	74018
24	0.551	0.016	0.52	0.58	74428.80	2891.57	67626	79452
26	0.636	0.016	0.59	0.66	78502.72	1979.19	73861	82669
28	0.724	0.022	0.68	0.76	82827.88	3123.51	76899	87766
30	0.817	0.017	0.78	0.86	86856.84	2069.58	82663	90288
32	0.919	0.018	0.88	0.95	91353.40	2630.55	85371	95612
34	1.011	0.028	0.97	1.06	94938.44	3060.75	88282	102151
36	1.123	0.029	1.07	1.19	100004.32	2764.49	94707	105113
38	1.240	0.033	1.18	1.30	103977.88	3086.72	98824	110148
40	1.342	0.037	1.28	1.42	107519.80	3637.69	99957	114997
42	1.464	0.027	1.39	1.52	112212.60	3192.65	106075	118548
44	1.597	0.040	1.52	1.67	115233.64	3221.49	109501	121623
46	1.719	0.045	1.62	1.80	118535.76	3537.88	111951	125069
48	1.849	0.055	1.74	1.95	120861.12	3976.51	110293	127368
50	2.000	0.062	1.84	2.13	127216.48	4114.49	120372	135073
52	2.119	0.055	2.00	2.20	129923.04	3841.80	120731	135491
54	2.299	0.065	2.11	2.45	133824.40	4051.75	123282	143042
56	2.417	0.058	2.30	2.54	136171.28	3623.44	129742	144099
58	2.560	0.061	2.40	2.66	139171.28	3944.68	129930	147636
60	2.740	0.067	2.59	2.88	144294.72	4178.43	136478	151523
62	2.846	0.054	2.74	2.95	144540.32	3046.71	135911	150749
64	3.033	0.067	2.90	3.20	148704.64	3872.89	140740	156719
66	3.232	0.092	3.07	3.39	152347.56	4599.68	144036	162516
68	3.457	0.082	3.31	3.60	157085.08	4043.41	149238	165330
70	3.560	0.081	3.38	3.68	156302.24	3707.40	149384	165515
72	3.756	0.097	3.60	3.97	160454.84	4547.77	152216	170379
74	3.977	0.089	3.78	4.22	164518.44	3546.08	157188	171084
76	4.212	0.121	3.95	4.39	168184.64	4011.74	157654	175070
78	4.437	0.098	4.24	4.61	171701.16	4370.90	162034	179837
80	4.616	0.101	4.39	4.83	173742.56	5627.97	159610	186329

Table 31: Trial 8B `RIPAL`Version 3: All problems have 4 dimensions, 8000 pieces. Solution time is explored as the number of IvP functions vary.

10.2.2 Data From Trial 8B - RIPAL Version 5

Table 32 contains the full results of RIPAL Version 5, for 25 randomly generated IvP problems, plotted in Figure 25. Time is shown in seconds.

	dim=4, pcs=8000							
	CPU Time (seconds)				Leaf Nodes			
ipfs	avg	sig	min	max	avg	sig	min	max
2	0.002	0.004	0.00	0.01	12.96	9.16	1	34
4	0.004	0.005	0.00	0.01	9.44	5.94	1	19
6	0.009	0.003	0.00	0.01	12.80	5.65	3	25
8	0.015	0.008	0.00	0.03	12.68	6.39	3	34
10	0.024	0.011	0.01	0.05	11.72	7.19	0	27
12	0.037	0.017	0.01	0.08	9.96	6.10	1	24
14	0.060	0.024	0.02	0.11	11.68	6.94	2	31
16	0.095	0.041	0.04	0.17	12.12	6.75	0	28
18	0.130	0.047	0.05	0.24	14.48	5.24	2	26
20	0.156	0.073	0.03	0.28	13.28	5.40	5	25
22	0.246	0.118	0.05	0.47	11.20	6.33	0	27
24	0.292	0.126	0.07	0.65	10.68	5.55	0	22
26	0.329	0.170	0.09	0.72	10.56	7.13	0	31
28	0.466	0.206	0.15	0.94	11.84	4.32	5	25
30	0.592	0.268	0.24	1.29	13.96	6.51	2	28
32	0.764	0.316	0.06	1.52	10.64	5.28	1	21
34	0.862	0.431	0.14	1.92	9.72	5.50	1	22
36	1.241	0.445	0.50	2.08	12.56	6.73	0	31
38	1.346	0.593	0.24	2.45	12.28	6.66	1	24
40	1.576	0.704	0.11	2.90	10.52	5.26	0	24
42	1.728	0.772	0.32	3.04	11.40	4.78	3	22
44	2.016	0.731	0.66	3.51	11.40	5.58	3	23
46	2.522	1.236	0.49	4.76	12.72	3.61	6	21
48	2.797	1.138	0.58	5.46	11.96	6.38	3	34
50	3.086	1.079	1.22	5.33	10.16	4.93	3	22
52	3.801	1.222	1.83	6.03	12.16	5.77	1	29
54	4.078	1.453	1.54	7.48	10.84	4.36	1	18
56	5.526	1.632	1.07	8.26	10.84	4.93	5	27
58	5.707	1.428	2.86	8.84	10.40	6.25	0	23
60	6.490	2.083	2.75	10.18	12.44	7.46	2	31
62	6.853	2.233	2.18	10.34	13.48	6.95	2	30
64	9.010	2.169	5.21	12.20	11.40	5.61	1	23
66	9.281	2.531	5.07	13.63	10.44	4.48	0	20
68	8.576	3.179	3.19	13.55	10.84	7.49	0	32
70	10.330	3.349	4.03	17.76	11.24	4.96	4	26
72	10.821	2.883	5.82	16.88	13.12	7.21	4	33
74	11.439	3.103	5.68	18.48	13.60	6.69	4	28
76	13.146	3.409	5.84	17.75	11.64	6.75	0	27
78	14.459	5.037	3.43	25.07	12.48	6.84	1	25
80	14.563	3.805	6.59	21.81	11.72	6.97	0	29

Table 32: Trial 8B `RIPAL` Version 5: All problems have 4 dimensions, 8000 pieces. Solution time is explored as the number of IvP functions vary.