Improved Cube Handling in Races: Insights with Isight

Axel Reichert (`mail@axel-reichert.de`)

2014-06-12

Abstract

You are an ambitious backgammon player? You like the KEITH count, but wish it were simpler? You like the KEITH count, but think it lacks accuracy? You like the KLEINMAN count or TRICE's Rule 62, but have no idea how their accuracy compares with the KEITH count? You like the KEITH count, but would prefer a method that gives winning percentages and works even in match play? You like BOWER's interpolation for the THORP count, but like the KEITH count better? You like the concept of the EPC, but have no idea how to calculate it for a complicated position? You have heard rumors about a "gold standard table", but do not want to calculate square roots over the board or distinguish between long and short races? You want a method for cube handling in races that can be used for positions with wastage? You want a method for cube handling in races that can be abused for mutual two-checker bear-offs?

Then you should read this article.

Contents

1 Introduction

Good cube handling in backgammon endgames is an important skill for ambitious players. Such situations come up frequently, but usually cannot be solved analytically by means that can be used legally over the board. Consequently, accurate heuristics that can be applied easily even when pressed for time (e. g. at tournaments with clocks) are needed. These typically consist of two things, an adjusted pip count and a decision criterion. An "adjusted pip count" does a much better job than a "straight" pip count (and, in most cases, than your gut feeling!) when judging your chances in a pure race (i. e., a race in which all contact has been abandoned). A "decision criterion" will tell you when to double, redouble, take, or pass.

If you are not familiar with backgammon theory for cube decisions, races, and pip counts, I recommend reading three articles by Tom Keith, Walter Trice, and Joachim Matussek:

- `http://www.bkgm.com/articles/CubeHandlingInRaces/`
- `http://www.bkgm.com/articles/EffectivePipCount/`
- `http://www.bkgm.com/articles/Matussek/BearoffGWC.pdf`

While `http://www.bkgm.com/` offers a wealth of information on various other backgammon topics, particularly these three should prepare you to make the most of the rest of this article.

After having a look into how adjusted pip counts and decision criteria work in general, we will then proceed to a more formal framework that will allow us to parametrize and optimize adjusted pip counts and the corresponding decision criteria. The outcome will be a new method (or, more precisely, several new methods) resulting in both less effort and fewer errors for your cube handling in races compared to existing methods (Thorp, Keeler, Ward, Keith, Matussek, Kleinman, Trice, Ballard etc.). Such a claim demands verification, hence we will look at results obtained with the new method and compare them to the existing ones for cube handling in races. Furthermore we will look at approximations of the effective pip count (EPC) or the cubeless probability of winning (CPW), which is a prerequisite of methods for match play. A summary concludes this article and recapitulates my findings for the impatient backgammon player who is always on the run, since the next race is in line online. Finally, the appendices contain several examples detailing the application of the new method and, for the readers not yet convinced of its merits, some rather technical remarks and further comparisons.

2 Adjusted Pip Counts

Pip counts need to be adjusted because straight (unadjusted) pip counts do a poor job when judging your racing chances. This is due to positional features that are neglected by a straight pip count, but might hurt you during bear-off:

- High stacks
- Gaps
- Crossovers
- More checkers to bear off

We will then deal with the question whether these features need to be punished or their lack should be rewarded. Why, for example, not give a "bonus" (by subtracting from the pip count) for an even distribution of checkers rather than a "penalty" (by adding to the pip count) for stacks/gaps? A similar problem is whether these positional features should be treated in an "absolute" fashion (punish/reward both players independently) or in a "relative" fashion (punish/reward one player only, employing numerical differences).

2.1 High Stacks (on Low Points)

Most players realize intuitively that high stacks, especially on low points, are not good during bear-off. Why is this so? In **figure 1** on the following page both White's and Red's pip counts are 5. If White rolls 6 and 5, his pip count will decrease by only 2, hence 9 pips are "wasted", a concept exploited by Walter Trice in his EPC. The higher the stack and the lower the point, the more overall wastage: Higher stacks mean more checkers to bear off with wastage, lower points mean more wastage per checker. In contrast, one checker borne off from the 5 point will at most waste a single pip. Wastage has a severe effect on winning chances, in fact in this example Red should double (even after White rolls a double) and White has an optional pass.

High stacks on low points hence call for a penalty to account for the wastage. But how many checkers make for a "high" stack? And how many points are "low"? Popular adjusted pip counts (Thorp, Keeler, Ward, Keith) penalize stacks on points 1, 2, and 3 with up to 2 additional pips. Some of these methods penalize only stacks higher than a particular "offset", e. g., Tom Keith recommends to "add 1 pip for each checker more than 3 on the 3 point". Likewise, the Keeler method uses a stack penalty of 1 and a stack penalty offset of 0 for point 1. We could even say that it uses a stack penalty of 0 for point 2 (since it does not penalize stacks there).

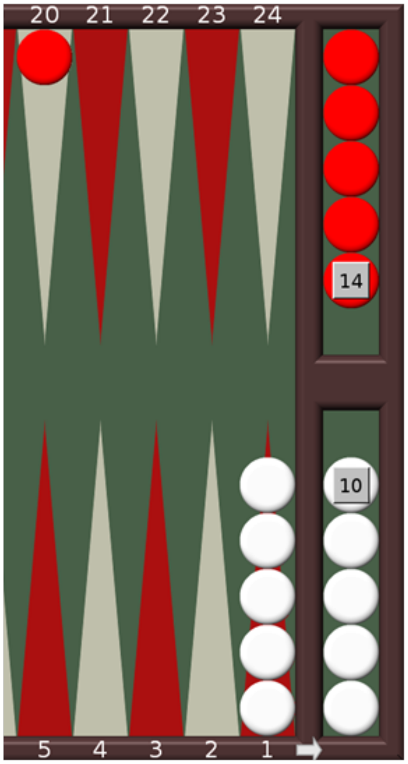
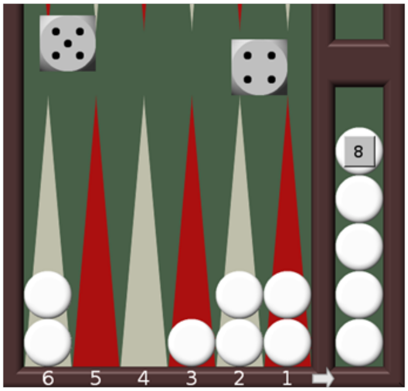


Figure 1: *Wastage by high stacks on low points*

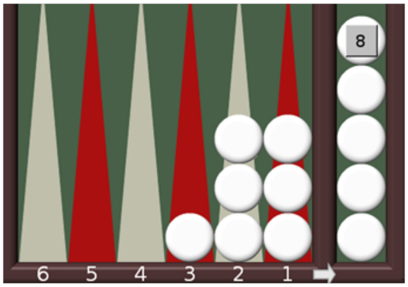
In a general framework, adjusted pip counts use a “stack penalty” and a “stack penalty offset” for each home board point.

2.2 Low Stacks (on High Points)

Having understood the basic concept of wastage, how does this apply to low stacks? Why might a gap (the most extreme form of a low stack) call for further adjustments of the pip count? This is because a gap usually leads to more wastage later on in the race, since it often forces you to move checkers from higher points to lower points. In **figure 2** on the next page, White rolls 5 and 4 and is forced to play 6/2 6/1: This bears off no checkers, but creates high stacks on low points, which lead to future wastage. Once on the lower points, these checkers will be penalized by the adjusted pip



(a) *Low stacks on high points . . .*



(b) *. . . give high stacks on low points*

Figure 2: *Future wastage by gaps*

counts. Penalizing gaps earlier in the race will account for future wastage. A second effect also warrants a penalty on gaps: Each time you roll a number corresponding to a gap, you will not be able to bear off a checker using this number. And sometimes you will not even be able to close a gap or smooth your distribution. Hence, in contrast to Douglas Zare (see `http://www.bkgm.com/articles/Zare/EffectivePipCount/index.html`) I believe there is strong evidence that gaps should be penalized, since they are potential ancestors of high stacks on low points (see above) or less checkers off (see below).

So a straight pip count needs to be adjusted for gaps on high points. But by how many pips? What makes for a "high" point? And what then constitutes a gap? Does a position with no checker on point 5 have a gap there at all if point 6 and higher points have already been vacated? The Keith method answers in the positive, since it simply penalizes empty space on points 4, 5, or 6 with a "gap penalty" of 1. The method by Paul Lamford even considers whether rolling the gap number allows for filling other gaps. In this case, the penalty is reduced or not even applied at all. In contrast, the other three well-known adjusted pip counts (Thorp, Keeler, and Ward) give a bonus (more on this later) for occupied points.

In a general framework, adjusted pip counts use a "gap penalty" (using a particular definition of "gap") for each home board point.

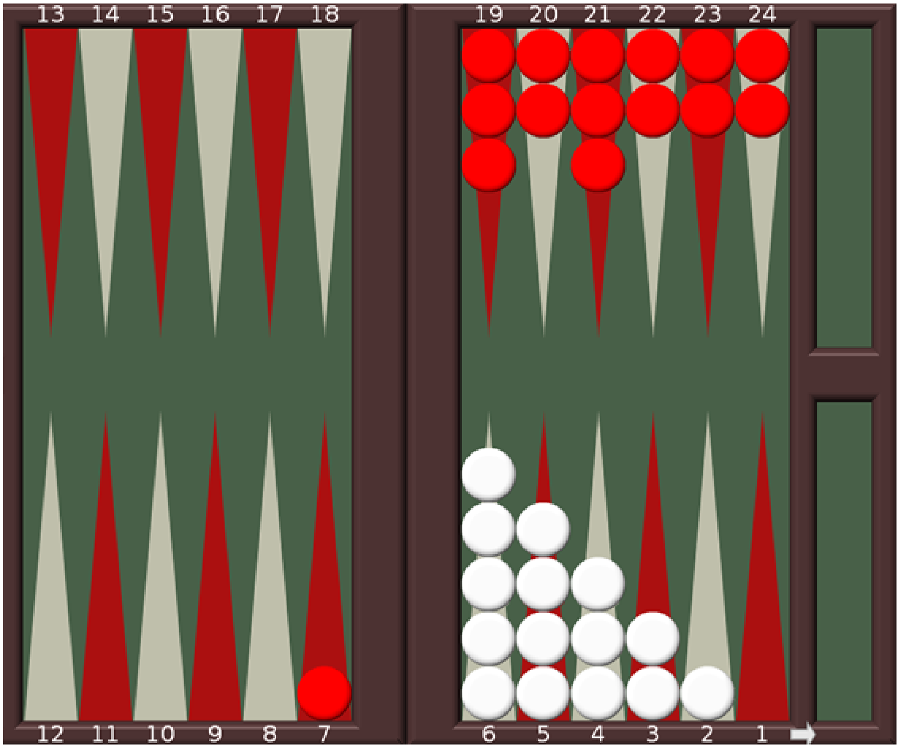


Figure 3: *More checkers out*

2.3 More Checkers Out

Races in which not all checkers have already been borne in are typically quite long regarding the pip count or have one player with a crunched home board plus one or two stragglers. In the latter case, one player still has to bring in the remaining checkers, while the other can already start to bear off. Adjusting the pip count further for checkers not yet in the home board thus seems an option. In **figure 3**, both White's and Red's pip counts are 70, but White's winning chances are about 66 %, according to GNU Backgammon. Red needs to move his last checker into his home board before he can start his bear-off, while White starts immediately.

Thus a straight pip count needs to be adjusted. But by how many pips? Of the four adjusted pip counts considered here, only the WARD method uses such a penalty: It adds half a pip for every checker outside the home board.

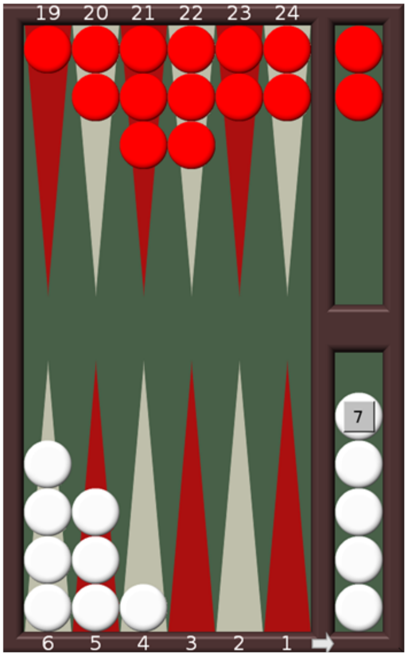


Figure 4: *Less checkers off*

One could also take into account how *far* the stragglers are behind. When counting crossovers, a checker that has just entered from the bar warrants more additional pips than a checker on your bar point.

In a general framework, adjusted pip counts use a "straggler penalty" or "crossover penalty" for checkers still outside the home board.

2.4 Less Checkers Off

Clearly the number of checkers still to bear off matters. In **figure 4**, both White's and Red's pip counts are 43, but White should double and Red should pass, according to GNU Backgammon. Red needs to bear off 13 checkers, while White has only 8 left.

So a straight pip count needs to be adjusted for additional checkers. But by how many pips? Thorp, Keeler, and Ward all penalize more checkers to bear off with 2 additional pips per checker.

In a general framework, adjusted pip counts use a "checker penalty" for this.

2.5 Bonus or Penalty?

So far, we have dealt only with penalties. How about the opposite concept, attributing bonus pips (subtracting from the straight pip count) for well-balanced positions without gaps or high stacks? There are two problems with this approach. First, *all* positions have wastage, which means *additional* pips you will have to roll on average to get your checkers off. You will never bear off a checker from point 4 with less than 4 pips. Second, with bonus pips the adjusted pip count might become zero or negative for very short races, which will render some of the decision criteria (see section 3 on the following page) rather dubious, e. g., how to add 10 % to a pip count of −3?

Despite these theoretical reservations, the Thorp, Keeler, and Ward methods all give a one pip bonus for additional occupied home board points. This is the only case in which a positional feature is awarded a bonus by one of the popular methods.

In a general framework, adjusted pip counts use a “point bonus” to account for occupied points in the home board.

2.6 Absolute or Relative?

Adjusting pip counts needs mental resources and time (important for tournaments played with clocks), so things should be kept simple. “Absolute” positional features need more effort than “relative” ones: It takes much more time to add 2 pips for each checker the player has on the board (like the Thorp method does) than to add 2 pips for each *additional* checker compared to the opponent (like the Keeler method does).

In a general framework, adjusted pip counts use 5 binary variables (“relative flags”) that control whether positional features are evaluated in an absolute or relative way, working independently on gaps, stragglers, crossovers, checkers still to bear off, and occupied points.

Let me give you some examples to explain this idea: With a relative flag for gaps, a gap on your point 4 would be penalized only if your opponent had no gap on his point 4. The same is true for the other points for which a gap penalty is applicable. With a relative flag for crossovers, you count the crossovers still needed for bear-in. Assume you still need 1 crossover, while your opponent needs 3. Hence his pip count will be increased by twice the crossover penalty. This also goes for a straggler penalty (with relative flag) used instead of a crossover penalty. And it would be perfectly fine for the same adjusted pip count to penalize checkers still on the board in an absolute fashion.

3 Decision Criteria

Once we have established an adjusted pip count, we need to review the various features of decision criteria for cube action. The most important are:

- Point of last take
- Doubling point
- Redoubling point

Some decision criteria additionally distinguish between a long and a short race.

3.1 Point of Last Take

Normally you need a reasonable lead in the adjusted pip count before offering a double. But what is your maximum lead, so that your opponent still has a take? Most methods add some fraction of the roller's count and add or subtract some more pips to determine this "point of last take". How large should this fraction be? And how many pips should further be added/subtracted? The KEITH method, for example, increases the roller's pip count by 1/7 (of the roller's count) and subtracts 2 pips. The THORP method uses only 1/10 of the roller's count, but adds 2 pips.

In a general framework, decision criteria use a "fraction" of the roller's pip count and some further "shift". Since the numerator for this fraction is typically 1, we will for convenience often refer only to the "denominator" instead of the complete fraction.

3.2 Doubling Point

Now that we know the point of last take for the opponent, it is time to think about the size of the doubling window. Most methods use a doubling window with a fixed size, e. g., WALTER TRICE's Rule 62 advocates a "doubling point" that is 3 pips shy of the point of last take. EDWARD THORP's criterion, which is also used for the KEELER and WARD method, states that the doubling point is 4 pips before the point of last take. The criterion for the KEITH method uses 2 pips.

In a general framework, decision criteria use a shift from the point of last take to determine the "doubling point".

3.3 Redoubling Point

For redoubles, the same procedure (i. e., applying a shift) is used by most methods, e. g., Walter Trice's Rule 62 uses a "redoubling point" 2 pips shy of the point of last take, the Keith method uses 1 pip, and Edward Thorp's criterion uses 3 pips. Of course, the redoubling point is between the doubling point and the point of last take.

In a general framework, decision criteria use a shift from the point of last take to determine the "redoubling point".

3.4 Long Race or Short Race?

Some decision criteria distinguish between a long and a short race. For example, Bill Robertie mentions a modification of Edward Thorp's decision criterion: For a long race (i. e., the roller has more than 30 pips) the pip count is increased by $1/10$ (of the roller's count). For shorter races this fraction is 0. Then a shift of 2 pips is used to determine the point of last take. Walter Trice's criterion not only uses different denominators (10 for long races, 7 for short races), but also different shift values (2 for long races, $-5/7$ for short races). He also defines 62 as the breakpoint between short and long races (hence the name Rule 62).

In a general framework, decision criteria use a "long race denominator" and a "long race shift" as well as a "short race denominator" and a "short race shift" for determining the point of last take. It is also necessary to define the "long/short breakpoint".

4 Multi-Objective Parameter Optimizations

In the previous sections we have seen that many adjusted pip counts and decision criteria fit into the same framework and thus some general parameters can be gathered and optimized so that cube handling errors are kept at a minimum.

4.1 Adjusted Pip Counts

Table 1 on the next page summarizes the parameters for adjusted pip counts. The parameter values are given for some existing methods and followed by the lower and upper limits specified for the optimization. The flag for "true gaps", if set, means that a vacated point counts as a gap only if there is at least one checker left on a higher point. Of course this definition could be changed to something more complicated, e. g., along the lines of Paul Lamford.

Table 1: *Parameters for adjusted pip counts*

Parameter	Existing methods				Optimization limits	
	Thorp	Keeler	Ward	Keith	Lower	Upper
Stack penalties						
1	1	1	2	2	0	3
2	0	0	1	1	0	2
3	0	0	0	1	0	2
Stack penalty offsets						
1	0	0	2	1	0	3
2	0	0	2	1	0	3
3	0	0	0	3	0	4
Gap penalties						
4	0	0	0	1	0	1
5	0	0	0	1	0	1
6	0	0	0	1	0	1
Straggler penalty	0	0	0.5	0	0	1
Crossover penalty	0	0	0	0	0	1
Checker penalty	2	2	2	0	0	2
Point bonus	1	1	1	0	0	1
Relative flags						
Gaps	0	0	0	0	0	1
Stragglers	0	0	0	0	0	1
Crossovers	0	0	0	0	0	1
Checkers	0	1	1	0	0	1
Points	0	1	1	0	0	1
True gaps	0	0	0	0	0	1

4.2 Decision Criteria

Table 2 on the following page summarizes the parameters for decision criteria. The lower and upper values defined for the optimization are also included. I did *not* distinguish between long and short races and thus needed only one denominator and one shift along with the (re)doubling point. More on this deliberate decision in appendix A.3 on page 37.

Table 2: *Parameters for decision criteria*

Parameter	Existing methods			Optimization limits	
	Thorp	Keith	Trice	Lower	Upper
Long/short breakpoint	30	0	62	–	–
Long race denominator	10	7	10	5	12
Long race shift	2	−2	2	−3	2
Short race denominator	–	–	7	–	–
Short race shift	2	–	$-5/7$	–	–
Doubling point	−4	−2	−3	−5	−2
Redoubling point	−3	−1	−2	−4	−1

4.3 Effort and Error

We now have identified 19 integer parameters describing an adjusted pip count and 4 integer parameters for a decision criterion. Each parameter has only few possible values, since we have specified reasonable upper and lower limits: The "relative flags", e. g., can take only 2 different values (0 or 1), while the "long race denominator" can take 8 different values, from 5 to 12. Using this framework, there are a total of

$$4 \cdot 3 \cdot 3 \cdot 4 \cdot 4 \cdot 5 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 35389440 \tag{1}$$

possible combinations for an adjusted pip count and

$$8 \cdot 6 \cdot 4 \cdot 4 = 768 \tag{2}$$

possible combinations for a decision criterion. We can also combine each adjusted pip count with each decision criterion, so overall this amounts to a "parameter space" of roughly $27 \cdot 10^9$ possible methods for cube handling in races. This parameter space is too big to be evaluated completely, hence we need a smart search technique, an optimization, to find within a finite time the "best" combination of an adjusted pip count with a decision criterion.

We have not yet clarified the meaning of "best" in the previous paragraph: Obviously, we want a method for cube handling that results in as few errors as possible. But we also want a method that requires as little effort as possible. In technical terms we are dealing with a multi-objective optimization problem. This normally involves a trade-off: The more effort, the less errors. It does not make sense to use a method that requires more effort *and* results in more errors. Such methods are "dominated" by others: The methods that result in the fewest errors for a given effort (or, vice versa, the methods that require the smallest effort for a given error) are on the "Pareto front".

In order to quantify the error, we choose a set of parameters for both, the adjusted pip count and the decision criterion, and test the resulting method on Tom Keith's database of endgame positions recorded on `http://www.fibs.com/`. For every position, the method calculates an adjusted pip count and recommends a cube action, which we compare with the correct cube action from the database (as determined by roll-outs with GNU Backgammon). Any wrong decisions (in terms of lost equity) can be summed up for all positions in the database and thus quantify the error resulting from a particular method.

Quantifying the effort is more complicated: It is not immediately clear whether an adjusted pip count using a stack penalty of 2 and a stack penalty offset of 2 needs less effort than an adjusted pip count using a stack penalty of 1 and a stack penalty offset of 1. What *is* clear, though, is that a gap penalty of 0 instead of 1 makes a method simpler, because there are fewer arithmetic operations. Likewise, lower checker penalties and lower stack penalties result in less effort. On the other hand, *higher* stack penalty *offsets* are easier (because the penalties themselves have to be applied less often). Of course, using "true gaps" means less effort, as does the use of all the "relative" flags. So for a given method/set of parameters I ignored the signs of all the penalties/bonuses applied by the adjusted pip count and summed up these absolute values for all positions in the endgame database. This number of total adjustments (measured in pips) was used to quantify the effort required by a particular method.

Having quantified our two objectives, we can look at some of their properties that will have a big influence on choosing a successful parameter optimization strategy. Technically speaking, our objective functions are nonlinear (e. g., doubling a gap penalty does not necessarily double the error), multimodal (have many local minima) and discrete (all parameters are integers). For these reasons, conventional gradient methods are likely to get stuck in local minima. Which is why we will start with a much better suited genetic algorithm that explores the parameter space to identify a promising region, which we will then narrow down with a more exhaustive search, technically a full factorial design of experiment (DOE), evaluating all of the remaining possible parameter combinations to find the optimum cube handling method. This ensures that a least in this region of the parameter space we do not overlook the global minimum among all the local minima of the highly nonlinear objective function.

5 Results

The dull work of trying to choose a set of 19 parameters such that our two objectives (effort and error) were minimized was done by Isight, a software solution for process automation and design exploration by Dassault Systèmes, the company I happen to work for. Essentially, what Tom Keith did for his method by trial and error was now done systematically and passed over to a software robot.

This exhaustive search found thousands of parameter combinations for adjusted pip counts and decision criteria that have a smaller error than Tom Keith's original method. This is not to diminish his outstanding work, in fact I consider it a great achievement to find a needle (his method with an error of 1262) in a haystack (the huge parameter space of 27 billion combinations). That a more rigorous approach (using sieves and magnets, to stay with the metaphor) would find more needles was to be expected. After about 5000 tested methods the optimization software found the following method (from now on termed "Isight method"):

5.1 Adjusted Pip Counts

For each player, start with a straight pip count and:

- add 1 pip for each additional checker on the board compared to the opponent;
- add 2 pips for each checker more than 2 on point 1;
- add 1 pip for each checker more than 2 on point 2;
- add 1 pip for each checker more than 3 on point 3;
- add 1 pip for each empty space on points 4, 5, or 6 (only if the other player has a checker on his corresponding point);
- add 1 pip for each additional crossover compared to the opponent.

You can see that empty space is penalized in a relative sense, which means that, e. g., your gap on point 4 is acceptable as long as your opponent also has a gap there. The same holds for the other high points, 5 and 6. This method does not penalize stragglers, but rather if one player has more crossovers than the opponent. It also penalizes less checkers off in a relative sense. It does not apply bonuses for positional features like occupied points. No sophisticated definition of "gap" is used, just plain empty space, no matter what goes on on higher points. Finally, we have the adjusted pip counts for both players.

5.2 Decision Criteria

Increase the count of the player on roll by 1/6.

- A player should double if his count exceeds the opponent's count by at most 6.
- A player should redouble if his count exceeds the opponent's count by at most 5.
- The opponent should take if the doubler's count exceeds his count by at least 2.

You will perhaps notice that this is very similar to the wording of the KEITH method. However, translating back to the terminology of section 3 on page 10, the decision criterion uses a doubling point 4 pips shy of the point of last take and a redoubling point 3 pips shy of the point of last take, hence the (re)doubling windows have the same size as with the THORP method, but are shifted: The "long race denominator" is 6 and the "long race shift" is -2 (in contrast to THORP's 10 and $+2$).

5.3 Effort and Error

Table 3 compares existing methods with the method found by Isight during the optimization. The Isight method has the smallest total error (1064), and, consequently, the smallest average error per decision (0.00688 equity). Not shown in the table, it also has the smallest probability of a wrong decision (7.1 %). So the Isight method performs best regarding our objective to minimize the error.

Even though the long list of items in section 5.1 on the previous page might suggest that the Isight method requires *more* effort than, say, the KEELER

Table 3: *Effort and error*

Method	Effort (pips)		Error (equity)	
	Total	Per player	Total	Per decision
THORP	2875000	27.9	3043	0.01969
KEELER	313000	3.0	2253	0.01458
WARD	330000	3.2	1838	0.01189
KEITH	379000	3.7	1262	0.00816
Isight	273000	2.6	1064	0.00688

Table 4: *Optimized parameters for adjusted pip counts*

Stack penalties for point			Stack penalty offsets for point			Gap penalties for point			Total effort (Pips)	Total error (Equity)
1	2	3	1	2	3	4	5	6		
2	1	1	2	2	3	1	1	1	273000	1064[a]
2	1	0	2	2	3	1	1	1	263000	1113
2	1	1	2	2	3	1	1	0	261000	1148
2	1	1	2	2	3	0	1	1	257000	1193
2	1	0	2	2	3	1	1	0	251000	1214
2	1	0	2	2	3	0	1	1	247000	1262

a. Isight method

or Keith methods (the first does not consider gaps, the second ignores less checkers off, both do not take crossovers into account), the opposite is true: Gap penalties applied only if the other player has a checker there and higher stack penalty offsets save a lot of arithmetic operations. They help so much in reducing the effort that in fact the Isight method results in *less* pip adjustments per player than the older methods.

From a multi-objective optimization point of view, you can see that the Thorp method (the first adjusted pip count in history?) requires more effort, but results in a higher error than the Keeler method, hence it is dominated by the latter. Later improvements such as from Jeff Ward or Tom Keith managed to bring down the error further, but at a cost: More effort is needed. The Isight method, however, has a lower effort *and* error and thus dominates all former methods.

During the optimization Isight literally found thousands of methods with an error smaller than for the Keith method. Of these, six are located on the Pareto front (remember, the trade-off between effort and error). They all use the decision criterion from section 5.2 on the preceding page, a crossover penalty of 1, and a checker penalty of 1. Neither straggler penalty, nor point bonus, nor true gaps are used. Gaps, crossovers, and checkers are penalized "relatively", i. e., compared to the opponent. The other parameters are summarized in **table 4**.

Comparing lines in this table that differ only by one parameter allows us to quantify the trade-off between effort and error: Failing to penalize stacks on point 3 costs about 60 points of equity (1113 versus 1064, 1214 versus 1148, and 1262 versus 1193). Not penalizing a gap on point 4 costs about 140 points (1193 versus 1064 and 1262 versus 1113), while not penalizing a

Table 5: *Race length and error*

Race length (up to pips)	Probability of error		Average error (equity)	
	Isight	Keith	Isight	Keith
10	0.1287	0.1362	0.03378	0.03780
20	0.1193	0.1175	0.01513	0.01561
30	0.0835	0.0845	0.00741	0.00824
40	0.0630	0.0694	0.00422	0.00623
50	0.0500	0.0588	0.00267	0.00409
60	0.0473	0.0602	0.00216	0.00363
70	0.0443	0.0597	0.00144	0.00258
80	0.0463	0.0510	0.00138	0.00164
90	0.0606	0.0583	0.00165	0.00155
100	0.0758	0.0737	0.00247	0.00217
110	0.0897	0.1004	0.00350	0.00309
120	0.0278	0.1250	0.00121	0.00555

gap on point 6 costs about 90 points (1148 versus 1064 and 1214 versus 1113). The simplest method has a total error of 1262 like the Keith method, but requires about 35 % less effort. Compared with the Isight method, it still saves about 10 % effort, but has an error about 20 % higher, a trade-off I do not consider to be worthwhile.

Let us now have a closer look at the probability of a wrong cube decision and the average error per cube decision with respect to the race length. All adjusted pip counts, including the Isight method, are considerably more error-prone for low pip counts (roughly for a race length below 30). For these, analytical solutions are still best, but unfortunately can be very difficult and time-consuming to calculate over the board. Having said this, **table 5** shows that the Isight method has a smaller average error (in terms of equity loss) than the Keith method for a race length up to 80 pips. From 80 pips up to 110 pips the Keith method is slightly more accurate, whereas in very long races (longer than 110 pips) again the Isight method has the edge. For races up to 10 pips, from 21 pips to 80 pips, and above 110 pips the Isight method is also less likely to result in a wrong cube decision.

5.4 Comments on Cub-offs

Even though for very short races both the likelyhood of a wrong cube decision and the size of the error increase considerably, it is interesting to push the envelope of the Isight method and assess its performance for “cub-offs”, a term coined by Danny Kleinman for bear-off positions with at most 2 checkers left per player. Overall, there are 729 cub-offs, and for 405 cub-offs the player on roll has a CPW between 50 % and 95 %.

The Isight method yields wrong decisions for about 12 % of these, a surprisingly small fraction. The average error per decision is only about 0.025. Unless you feel very confident about memorizing the complete cub-off tables available on the internet (`http://www.bkgm.com/articles/Koca/CuringYourShortBearoffBlues.html`), or, alternatively, you feel very confident about doing the math over the board (remember, without pencil and paper, but including potential redoubles . . .), it thus might be a reasonable approach to (ab)use the Isight method even for cub-offs.

6 Alternative Approaches

6.1 EPC Approximations

A concept quite different from adjusted pip counts is the EPC invented by Walter Trice, see the article mentioned in section 1 on page 3. In brief, the EPC of a particular position is the average number of rolls needed to bear off completely, multiplied by the average number of pips in a roll (which is $^{49}/_{6}$).

The problem with this approach is to determine the average number of rolls, since we do not have exact EPCs over the board. While this can easily be estimated for “rolls” positions (ace-point stacks) or low-wastage “pips” positions, in general, for the many positions that fall in between, the procedure is far from easy: It requires remembering the wastage for quite a number of cases, a good deal of experience and feeling for the position, or “synergy” wizardry (see Douglas Zare’s article mentioned in section 2.2 on page 5, which unfortunately does not explain the details). To me, the whole concept, while very elegant from a mathematical point of view, seems rather fuzzy and not very well suited for a strict algorithmical solution, which is much more to my liking.

But perhaps could such an algorithm be found by assessing positional features and adjusting for them (as explained in section 2 on page 4), so that we do not end up with some arbitrary adjusted pip count, but rather with something approximating the EPC as closely as possible? This approach seems to be quite promising and was Joachim Matussek’s idea, see his article mentioned in section 1 on page 3. He applies stack penalties (for all points from 1 to 6), adds a further pip if any stack is at least 4 checkers high and yet another pip if any stack is at least 6 checkers high. His method also uses a “synergy” effect, i. e., it subtracts a bonus of 1 pip if point 1 is occupied, but at least 3 other points are occupied as well. Another distinct feature is the use of a constant penalty applied always, since *every* position will waste at least a couple of pips. He uses 4.5 pips for this purpose. His method results in an average EPC error per player per position of 1.70 pips (for each position in the database EPCs for *both* players need to be approximated).

So I augmented Tom Keith’s database of endgame positions with EPCs as determined by GNU Backgammon (which fortunately very well supports such automation of tasks by scripting). Then I changed the objective function of the optimization problem: Now I did not want to minimize the total cube error, but rather the difference between the adjusted pip count and the correct EPC from the database. I also had to introduce additional parameters for the synergy effect and the constant penalty. Then I could start the optimization process. This is what Isight found as an EPC approximation:

For each player, start with a straight pip count and:

- add 5 pips;
- add 2 pips for each checker on point 1;
- add 1 pip for each checker on point 2;
- add 1 pip for each checker on point 3;
- subtract 1 pip if point 1 and at least 3 other points are occupied.

This method allows to estimate the EPC for a player’s position with an average error of only 1.04 pips, hence it has a smaller total error *and* requires less effort than Joachim Matussek’s method, which uses non-integer penalties for *all* home board points.

While an approximation for EPCs is certainly nice to have, our main purpose is cube handling in races, so we need to combine our “adjusted pip count” (which by now has become an EPC approximation) with a suitable decision criterion. Indeed there is one, somewhat hidden in Walter Trice’s book, with a denominator of $^{49}/_{6}$ and a shift of -3. Combining this criterion with the EPC approximation from above, we get a total error of 1909 (now a total cube error in terms of equity, not an EPC error in terms of pips), much worse than the Keith method (1262) or the Isight method (1064) from section 5 on page 15. The question why this result is so bad needs some investigation.

The problem with EPCs, even if you have an algorithm to estimate them over the board, is that Walter Trice’s decision criterion is very sensitive to the EPCs, since it has a doubling point only 2 pips shy of the point of last

Table 6: EPC *approximations with* WALTER TRICE*'s decision criterion*

EPC approximation	EPC error per player (Pips)	Total cube error (Equity)
Isight	1.04	1909
MATUSSEK	1.70	1498
Exact ±1.5	0.75	1294
Exact ±1.3	0.65	1067
Exact ±1.0	0.50	800
Exact	0.00	401

take. You can imagine that being 2.5 pips off with your EPC approximation can easily be the difference between "No double, take" and "Redouble, pass". This reasoning could be confirmed by hard numbers: I took the exact EPCs from the database and added/subtracted uniformly distributed random numbers. The resulting "EPC approximations" were combined with WALTER TRICE's decision criterion to determine the cube action. As usual, the total error was summed up for all positions in the database, see **table 6**.

The results are disappointing: In order to get a total cube error comparable to the Isight method (1064) from section 5 on page 15, we need an EPC approximation with an average EPC error of about 0.65, which is probably very difficult to achieve. However, assuming we had the exact EPCs, WALTER TRICE's criterion gives extremely accurate cube decisions. Unfortunately, it does not work very well with the EPC approximations we have found so far. This is an area with room for improvements in the future. Maybe someone will find an approximation that is accurate enough to be useful in combination with WALTER TRICE's criterion, which is definitely not the culprit, see the total error of only 401!

An interesting detail is that JOACHIM MATUSSEK's EPC approximation, though worse than Isight's, results in a *smaller* total cube error. We can take this as a hint that adjusted pip count and decision criterion need to be matched. However, JOACHIM MATUSSEK does not employ WALTER TRICE's criterion (his article does not deal with cube decisions at all), but rather uses the EPC approximation he found to approximate the CPW. This leads us to yet another approach.

6.2 CPW Approximations

If we had them, exact CPWs in combination with percentages defining the doubling/redoubling window would be even more accurate than using exact

EPCs with Walter Trice's criterion. According to Tom Keith the total cube error would be 201. He also mentions:

> Unfortunately, it is usually harder to accurately estimate CPW than it is to make correct cube plays, so this method is really more of academic interest.

Fortunately, Tom Keith is not right. Let us see how CPW approximations work.

The formula Joachim Matussek uses to approximate the cubeless probability p of winning (in percent) for a race length l (roller's pip count) and a roller's lead Δl is

$$p(l, \Delta l) = 50\,\% + v(l) \cdot \left(\Delta l + \frac{49}{12}\right) \ , \tag{3}$$

with v being the value of a pip (depending on the roller's pip count). This seems reasonable, since it has the nice property of giving 50 % winning chances if the roller is about 4 pips behind (half a roll, 49/12), a CPW we know for some decades to be true for a wide range of race lengths. Both roller's and non-roller's pip counts can be (and usually are) adjusted. Complications arise when the value of a pip needs to be factored in. Joachim Matussek gives a table for this, which needs further adjustments for positions in which at least one player has a "rolls" position rather than a "pips" position.

Since I did not want to memorize and adjust values from a look-up table, I thought about a table published by Tom Keith showing the CPW directly as a function of race length and lead, `http://www.bkgm.com/rgb/rgb.cgi?view+1203`. My idea was to approximate it using a linear regression

$$p(l, \Delta l) = b - \frac{l}{d_\mathrm{C}} + v \cdot \Delta l \ , \tag{4}$$

for which the base probability b, the CPW denominator d_C, and the (now constant) value of a pip v needed to be determined. Initial tests of fitting Tom Keith's table with this approach were promising. Using optimization software, this was again an easy task. With the CPW error (absolute difference between the correct CPW as rolled out by GNU Backgammon and its approximation) as an objective to minimize, Isight found the following method:

$$p = 77 - \frac{l}{4} + 2\Delta l \tag{5}$$

This CPW approximation has an average CPW error of 4.13 percentage points. The next optimization problem for Isight was to find parameter values (i. e.,

Table 7: *Parameters for* CPW *approximations*

Parameter	Lower	Upper	Unit
Base probability b	50	100	%
CPW denominator d_{CPW}	2	10	–
Pip value v	1	6	%
Doubling point	65	75	%
Redoubling point	69	75	%
Point of last take	75	80	%

percentages) for the doubling point, the redoubling point, and the point of last take such that combined with this CPW approximation the total error for the cube decisions was minimized. The parameters and their limits for these two runs are shown in **table 7**. The limits were chosen using a mixture of guesswork, gut feeling, and old backgammon lore. They should reasonably bracket the optimum. The first three parameters were used in the first run for the optimization of the CPW approximation, the second three parameters were used in the second run for the optimization of the (re)doubling window. Isight found that with a doubling point of 68 %, a redoubling point of 70 %, and a take point of 76 % this CPW approximation gives a total cube error of 1264. This is roughly on par with the KEITH method (1262), but worse than the Isight method (1064) from section 5 on page 15.

So I started another run, this time as a multi-objective optimization with both the CPW error *and* the total cube error to be minimized *simultaneously*, using *all* parameters from table 7. The parameter limits were unchanged. Isight then found the following CPW approximation for a roller's pip count l and a roller's lead Δl:

$$p = 80 - \frac{l}{3} + 2\Delta l \tag{6}$$

The (CPW-based) decision criterion uses the following percentages for the doubling and redoubling window to determine the cube action:

- $p < 68$: No double, take.
- $68 \leq p < 70$: Double, take.
- $70 \leq p \leq 76$: Redouble, take.
- $p > 76$: Redouble, pass.

Even though (due to the multi-objective nature of the optimization task) this method compromises a little bit on average CPW error (4.26 percentage points in comparison to the former 4.13), it does much better than equation (5) when it comes to cube errors. It fact, it gives *exactly the same* total cube error of 1064 as the Isight method. Even the decimal places (omitted here) matched perfectly. Could this be pure chance? Probably not, and we will see why in the next subsection.

6.3 Methods for Matches

The equations for CPW approximation somehow reminded me on Chuck Bower's interpolation for the Thorp method. In case you are not familiar with it, this is how it works (his article can be found at `http://www.bkgm.com/rgb/rgb.cgi?view+158`): If a Thorp count T of -2 corresponds to the doubling point (say 70 %) and a Thorp count T of $+2$ corresponds to the point of last take (say 78 %), then a simple linear interpolation will give a CPW approximation:

$$p = 74 + 2T \tag{7}$$

The Isight method combines the adjusted pip count from section 5.1 on page 15 with the decision criterion from section 5.2 on page 16. This results in a total cube error of 1064. If now the same adjusted pip count is combined with the decision criterion from the previous subsection, we also get a total cube error of 1064. Hence I had a strong suspicion that there might be a general way to transform a decision criterion based on (re)doubling points and the point of last take into a decision criterion based on percentages and the CPW. A criterion of the latter form has the big advantage that it is suited for match play, in which the doubling window might be far away from the usual values for a money session. And indeed, there is a correspondence, and the reasoning and the transformation are quite easy, since the decision criteria have all been parametrized earlier.

You might recall from section 3.1 on page 10 that we get the point of last take by increasing the roller's pip count l by a fraction of the roller's pip count (determined by the "long race denominator" d_l) and adding a further shift s. If the non-roller, who has a pip count of $l + \Delta l$, is precisely at this point of last take with a CPW of p_t (e. g. 78 %), we can state the following implication:

$$l + \frac{l}{d_\text{l}} + s = l + \Delta l \ \Rightarrow p = p_\text{t} \tag{8}$$

Since the doubling point is s_d pips shy of the point of last take (s_d being *negative*), which corresponds to a CPW of p_d (e. g. 70 %), we can state the following implication:

$$l + \frac{l}{d_\mathrm{l}} + s + s_\mathrm{d} = l + \Delta l \;\Rightarrow p = p_\mathrm{d} \tag{9}$$

After subtracting l from both sides of these two equations, linear interpolation gives the following “transformation formula”:

$$p = p_\mathrm{d} - (p_\mathrm{t} - p_\mathrm{d}) \frac{\Delta l - \left(\frac{l}{d_\mathrm{l}} + s + s_\mathrm{d}\right)}{s_\mathrm{d}} \tag{10}$$

Let us try this formula on the Isight decision criterion presented in section 5.2 on page 16. The long race denominator d_l is 6, the shift s is -2, and the doubling point s_d is -4. With a CPW of 70 % at the doubling point and 78 % at the point of last take this results in

$$p = 70 - (78 - 70) \frac{\Delta l - \left(\frac{l}{6} - 2 - 4\right)}{-4} \;, \tag{11}$$

which after a little bit of algebra turns out as

$$p = 82 - \frac{l}{3} + 2\Delta l \;. \tag{12}$$

Does not this look familiar yet? Not quite? Well, using this CPW approximation with a doubling window from 70 % to 78 % will by design give *exactly* the same cube decisions as using 80 instead of 82 for the constant term in the CPW approximation with a doubling window from 68 % to 76 %. All numbers will just be shifted by 2 %. So finally we are back to

$$p = 80 - \frac{l}{3} + 2\Delta l \tag{13}$$

with a (re)doubling window of 68 %, 70 %, and 76 %. It will give us a total cube error of 1064 as well, but has a slighly smaller CPW error than equation (12) and is thus better suited for match play, in which the percentages used for the (re)doubling window will have to be adapted based on score. The transformation formula thus has provided a very nice connection between the Isight solutions for the two (originally unrelated) optimization problems.

If my framework for adjusted pip counts and decision criteria can be seen as a generalization of Tom Keith’s work, my transformation formula can be seen as a generalization of Chuck Bower’s work: Once you have an adjusted pip count and a decision criterion using a point of last take and a (re)doubling window of a particular size, you can easily convert it into an

Table 8: CPW *approximations with transformation formula*

Method	b	d_C	v	Doubling point	Redoubling point	Point of last take	Total error	Average CPW error
	%	–	%	%	%	%	–	%
Thorp[1]	74	5	2	70	72	78	4433	5.72
Keith[2]	86	$7/4$	4	70	74	78	1262	10.09
Matussek[3]	90	$49/24$	4	70	74	78	1498	10.86
Matussek[4]	90	2	4	70	74	78	1485	10.75
Isight	77	4	2	68	70	76	1264	4.13
	80	3	2	68	70	76	1064	4.26
	82	3	2	70	72	78	1064	4.54
	85	2	3	68	71	79	1049	8.27

1. Without Bill Robertie's enhancement (no long/short race distinction)

2. Necessary rounding down not represented by parameter values

3. Combined with Walter Trice's decision criterion for EPC approximations

4. Like 3., but rounded value used for denominator

equivalent method working with percentages. Both methods will by design result in exactly the same cube action in a money session, however the latter will also work for match play. This is important for tournament players, indeed I was specifically asked by former German champion Tobias Hellwag whether I had such a method in my theory bag. Now I do ...

In case you are flirting with applying the transformation formula to your favorite Thorp or Keith method (as an "exercise for the reader"), a word of warning is in place: Of course this is possible, but for both methods it is not as straight forward as for the Isight method. The Thorp method (at least in its version enhanced by Bill Robertie) distinguishes between long and short races, while the Keith method does some rounding down during the definition of the point of last take. These features complicate things a little bit, hence I cheated somewhat with the numbers in **table 8**: The results for the CPW-based version of the Thorp method hold for his *original* formulation, i. e., without distinction between long and short races. For the Keith method you need to know where the rounding takes place:

$$p = 86 - 4 \cdot \left\lfloor \frac{l}{7} \right\rfloor + 4\Delta l \tag{14}$$

By the way, as good as the Keith method is, for match play its average CPW error is surprisingly high. This perhaps explains Tom Keith's quote from the beginning of section 6.2 on page 21. The same holds for Joachim Matussek's method, which was meant as an EPC approximation. As a detail, this method slightly benefits both in terms of total cube error and CPW error, if instead of the exact value for the CPW denominator ($^{49}/_{24}$, resulting from the transformation formula) a rounded value of 2 is used.

The methods found by Isight all do a very good job regarding the total cube error. The first three of them should be familiar from this and the previous subsection. The last one was not yet mentioned, it gives the smallest total cube error of all CPW-based methods found by Isight, however its CPW is much higher than for the other ones, which is bad for match play.

7 Discussion

This paper presented a general, parametrized framework for adjusted pip counts and decision criteria for cube handling in pure races without contact. The positional features considered for adjusted pip counts were checkers already borne off, high stacks on low points, gaps on high points, and crossovers needed before the bear-off starts. For decision criteria a distinction between long and short races was considered, but turned out to be unnecessary, since long races are rare. The point of last take was determined using a denominator and a further shift. In turn, doubling and redoubling point were shifted from the point of last take.

Having the framework in place, lower and upper limits for all the parameters were specified. Two objectives were defined: The total cube error should be minimized (tested against a database containing the correct cube actions for around 50000 real life endgame positions), and the total effort should be minimized (quantified using the total number of pips added/subtracted by the adjustments). Several multi-objective optimizations were done using the software solution Isight. First, an adjusted pip count and a decision criterion were found. The resulting method for cube handling in races has both a lower effort *and* error than existing methods. It can be reasonably used even for cub-offs. It is crucial that the decision criterion be valid for positions *with* wastage in order to match an adjusted pip count successfully. Second, a method for approximating EPCs was found, with both a lower effort *and* EPC error than existing methods. This method is not suitable for cube handling. Third, a method for approximating CPW was found, with a decision criterion that resulted in *exactly* the same cube action as the method found before.

Indeed, a connection between these two methods was found: The transformation formula describing it can be used to convert any decision criterion based on a point of last take and a (re)doubling window into an equivalent method working with percentages. This allows for the use in match play, for which the percentages will have to be adapted based on score.

8 Summary

- To approximate EPCs, start with a straight pip count for each player.
    - Add 5 pips.
    - Add 2 pips for each checker on point 1.
    - Add 1 pip for each checker on point 2.
    - Add 1 pip for each checker on point 3.
    - Subtract 1 pip if point 1 and at least 3 other points are occupied.
- To decide on cube action, start with a straight pip count for each player.
    - Add 1 pip for each additional checker on the board compared to the opponent.
    - Add 2 pips for each checker more than 2 on point 1.
    - Add 1 pip for each checker more than 2 on point 2.
    - Add 1 pip for each checker more than 3 on point 3.
    - Add 1 pip for each empty space on points 4, 5, or 6 (only if the other player has a checker on his corresponding point).
    - Add 1 pip for each additional crossover compared to the opponent.
    - With your adjusted pip count l and your lead Δl (can be negative),

      $$p = 80 - \frac{l}{3} + 2\Delta l$$

      is your approximated CPW. The cube action for money is:
    - $p < 68$: No double, take.
    - $68 \leq p < 70$: Double, take.
    - $70 \leq p \leq 76$: Redouble, take.
    - $p > 76$: Redouble, pass.
    - Adapt (re)doubling window according to match equity calculations in tournaments.

A Appendix

A.1 Examples

After so much generalization, parametrization, design space exploration, multi-objective optimizations, equations, interpolation, transformations, and approximations of various acronyms it is high time to apply the Isight method to some examples. Except for one position ("pips" versus "rolls"), the following examples have all been taken randomly from Tom Keith's database of endgame positions played on FIBS. We will apply the adjusted pip count and the decision criterion based on the CPW approximation to all of them. In addition, the summary table below each figure will also give the values from the EPC approximation. The outcome will be compared with the correct results from GNU Backgammon.

The purpose of these examples is *not* the verification of the methods, which has already been done by testing them on the full database, and the results of this have been presented in the main part of this paper. Instead, these examples should rather be seen as exercises in applying the methods and as a way to resolve any ambiguities (hopefully few) still present in their formulation.

In **figure 5** on page 31 the straight pip count is 86 versus 90. Both players have to bear off the same number of checkers, so no checker penalty is applied. No player has more than 2 checkers on his point 1, more than 2 checkers on his point 2, or more than 3 checkers on his point 3, so no penalties need to be applied for stacks. No player has gaps on his points 4, 5, or 6, so no penalties need to be applied for gaps. White has 4 checkers with 1 crossover and 1 checker with 2 crossovers, in total 6 crossovers. Red has 5 checkers with 1 crossover each, in total 5 crossovers. Hence White gets 1 pip penalty. The adjusted pip count is 87 versus 90, so the race length is 87 pips and the lead is 3 pips. This results in a CPW for White of $80 - {}^{87}\!/_{3} + 2 \cdot 3 = 57\,\% < 68\,\%$. The cube action is thus "No double, take". A rollout with GNU Backgammon confirms the cube action, but gives a CPW of 65.6 %.

In **figure 6** on page 31 Red has borne off 2 checkers less than White, so Red gets 2 pips penalty. Red also has 5 checkers on point 2, 3 more than "allowed", so Red gets another 3 pips penalty. White has 5 checkers on point 3, 2 more than the "allowed" 3, so White gets 2 pips penalty. White has a gap on point 4, whereas Red does not, so White gets 1 pip penalty. On the other hand, Red has a gap on point 6, whereas White does not, so Red gets one pip penalty, too. Red has 1 additional crossover and so gets another 1 pip penalty. The adjusted pip count is 53 versus 62. White's lead is 9 pips. His CPW is

$80 - {}^{53}/_3 + 2 \cdot 9 \approx 79.7\,\% > 76\,\%$. The cube action is thus “Redouble, pass”. This is confirmed by GNU Backgammon, which reports a CPW of 80.2 %.

In **figure 7** on page 32 White gets 1 pip penalty due to 12 checkers off against 13 for Red. There are no stacks, but gaps: 1 pip gap penalty for Red on point 4, 1 pip gap penalty for White on point 5, no gap penalty on point 6. No crossovers left. The adjusted pip counts become 10 versus 11, White’s lead is 1 pip. His CPW is $80 - {}^{10}/_3 + 2 \cdot 1 \approx 79.7\,\% > 76\,\%$. The cube action is thus “Redouble, pass”. GNU Backgammon confirms this and reports a CPW of 84 %.

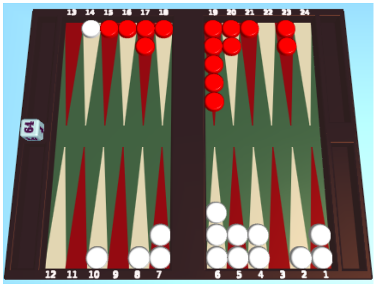
In **figure 8** on page 32, the Isight method is abused for a cub-off. White gets an additional pip for its 2 checkers left against 1 checker for Red. No stack penalties, but 1 pip gap penalty for Red on point 4 (since White has occupied its point 4). No crossover penalties. The adjusted pip count is 7 versus 4. The race length is 7, White’s lead is -3. His CPW is $80 - {}^{7}/_3 + 2 \cdot (-3) \approx 71.7\,\% \geq 70\,\%$. Since $71.7\,\% \leq 76\,\%$, the cube action is “Redouble, take”. GNU Backgammon also reports “Redouble, take”, but a CPW of only 63.9 %.

In **figure 9** on page 33, only Red gets 1 pip penalty, since White has borne off 1 more checker. No penalties for stacks, gaps, or crossovers. The adjusted pip count becomes 52 versus 60. White’s lead is 8 pips. His CPW is $80 - {}^{52}/_3 + 2 \cdot 8 \approx 78.7\,\% > 76\,\%$. The cube action becomes “Redouble, pass”, confirmed by GNU Backgammon, which reports a CPW of 80.5 %.

In **figure 10** on page 33, an extreme “pips versus rolls” position, the straight pip count (30 versus 15) is highly misleading. Red gets 4 pips penalty for his fewer checkers borne off. Red gets 6 pips penalty for his 3 checkers “too many” on point 1. Red also gets 3 pips penalty for his 3 checkers “too many” on point 2. Finally, Red gets 3 pips penalty for his gaps on points 4, 5, and 6, all of which White has occupied on his side of the board. The adjusted pip count becomes 30 versus 31. White *leads* by 1 pip! His CPW is $80 - {}^{30}/_3 + 2 \cdot 1 = 72\,\% \geq 68\,\%$. The cube action is “Double, take”. GNU Backgammon confirms the cube action and gives a CPW of 67.7 %.

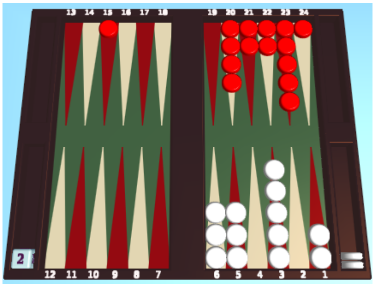
In **figure 11** on page 34 White gets 2 pips penalty for his third checker on point 1. He also gets 1 pip penalty for his third checker on point 2. Red accordingly gets 4 pips penalty for his stack on point 2 (4 checkers “too high”, 6 instead of 2). No penalties for crossovers or gaps (because *both* players have vacated their points 4, 5, and 6, there are no “relative” gaps). The adjusted pip count is 18 versus 18, White’s lead is 0. His CPW is $80 - {}^{18}/_3 + 2 \cdot 0 = 74\,\% > 68\,\%$. The cube action becomes “Double, take”, confirmed by GNU Backgammon, which reports a CPW of 74.7 %.

Figure 12 on page 34 concludes this appendix with two example positions, which can be used to test your understanding of the methods. For these, only the final results are presented, but no complete summary tables. Good luck!



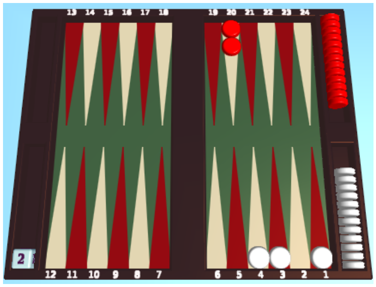
	White	Red
Straight	86	90
Penalties		
Checkers	0	0
Stack on 1	0	0
Stack on 2	0	0
Stack on 3	0	0
Gap on 4	0	0
Gap on 5	0	0
Gap on 6	0	0
Crossovers	1	0
Adjusted	87	90
Prediction (Isight method)		
CPW (%)	57.0	43.0
Cube action	No double	Take
EPC	95	97
Solution (GNU Backgammon)		
CPW (%)	65.6	34.4
Cube action	No double	Take
EPC	95.5	98.3

Figure 5: *Long race*



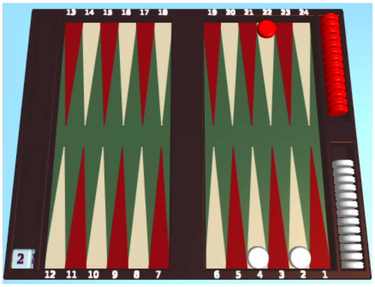
	White	Red
Straight	50	55
Penalties		
Checkers	0	2
Stack on 1	0	0
Stack on 2	0	3
Stack on 3	2	0
Gap on 4	1	0
Gap on 5	0	0
Gap on 6	0	1
Crossovers	0	1
Adjusted	53	62
Prediction (Isight method)		
CPW (%)	79.7	20.3
Cube action	Redouble	Pass
EPC	63	68
Solution (GNU Backgammon)		
CPW (%)	80.2	19.8
Cube action	Redouble	Pass
EPC	62.0	68.5

Figure 6: *Medium-length race*



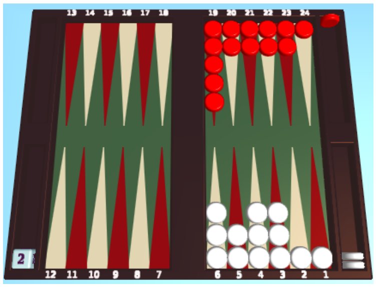
	White	Red
Straight	8	10
Penalties		
Checkers	1	0
Stack on 1	0	0
Stack on 2	0	0
Stack on 3	0	0
Gap on 4	0	1
Gap on 5	1	0
Gap on 6	0	0
Crossovers	0	0
Adjusted	10	11
Prediction (Isight method)		
CPW (%)	79.7	20.3
Cube action	Redouble	Pass
EPC	16	15
Solution (GNU Backgammon)		
CPW (%)	84.0	16.0
Cube action	Redouble	Pass
EPC	15.6	15.6

Figure 7: *Short race*



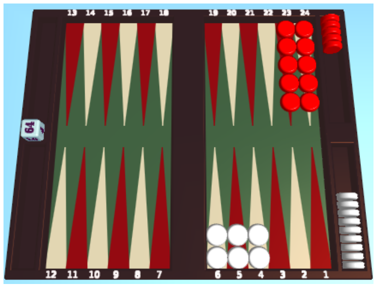
	White	Red
Straight	6	3
Penalties		
Checkers	1	0
Stack on 1	0	0
Stack on 2	0	0
Stack on 3	0	0
Gap on 4	0	1
Gap on 5	0	0
Gap on 6	0	0
Crossovers	0	0
Adjusted	7	4
Prediction (Isight method)		
CPW (%)	71.7	28.3
Cube action	Redouble	Take
EPC	12	9
Solution (GNU Backgammon)		
CPW (%)	63.9	36.1
Cube action	Redouble	Take
EPC	11.1	8.2

Figure 8: *Cub-off*



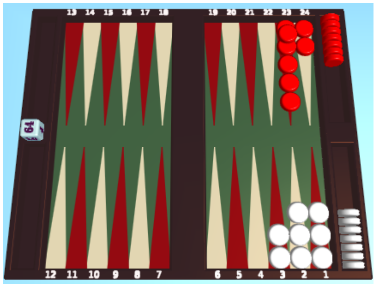
	White	Red
Straight	52	59
Penalties		
Checkers	0	1
Stack on 1	0	0
Stack on 2	0	0
Stack on 3	0	0
Gap on 4	0	0
Gap on 5	0	0
Gap on 6	0	0
Crossovers	0	0
Adjusted	52	60
Prediction (Isight method)		
CPW (%)	78.7	21.3
Cube action	Redouble	Pass
EPC	62	69
Solution (GNU Backgammon)		
CPW (%)	80.5	19.5
Cube action	Redouble	Pass
EPC	60.6	67.6

Figure 9: *"Pips" position*



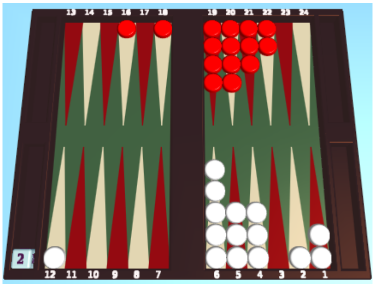
	White	Red
Straight	30	15
Penalties		
Checkers	0	4
Stack on 1	0	6
Stack on 2	0	3
Stack on 3	0	0
Gap on 4	0	1
Gap on 5	0	1
Gap on 6	0	1
Crossovers	0	0
Adjusted	30	31
Prediction (Isight method)		
CPW (%)	72.0	28.0
Cube action	Double	Take
EPC	35	35
Solution (GNU Backgammon)		
CPW (%)	67.7	32.3
Cube action	Double	Take
EPC	36.3	36.2

Figure 10: *"Pips versus rolls" position*

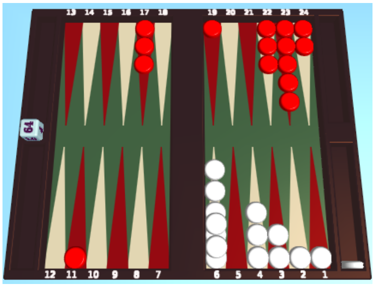


	White	Red
Straight	15	14
Penalties		
Checkers	0	0
Stack on 1	2	0
Stack on 2	1	4
Stack on 3	0	0
Gap on 4	0	0
Gap on 5	0	0
Gap on 6	0	0
Crossovers	0	0
Adjusted	18	18
Prediction (Isight method)		
CPW (%)	74.0	26.0
Cube action	Double	Take
EPC	31	29
Solution (GNU Backgammon)		
CPW (%)	74.7	25.3
Cube action	Double	Take
EPC	29.9	30.0

Figure 11: *"Rolls" position*



	White	Red
Prediction (Isight method)		
CPW (%)	67.7	32.3
Cube action	No redouble	Take
Solution (GNU Backgammon)		
CPW (%)	69.0	31.0
Cube action	No redouble	Take



	White	Red
Prediction (Isight method)		
CPW (%)	81.0	19.0
Cube action	Double	Pass
Solution (GNU Backgammon)		
CPW (%)	83.5	16.5
Cube action	Double	Pass

Figure 12: *Two quiz positions*

A.2 Combinations of Counts and Criteria

The parametrization of adjusted pip counts and decision criteria all using the same (or a very similar) framework allowed to test a lot of combinations. During my hunt for an improved method for cube handling I implemented the following adjusted pip counts for testing:

- Straight (not adjusted)
- Thorp
- Keeler
- Ward
- Keith
- Isight (adjusted pip count, EPC approximation, and CPW approximation)
- Matussek (EPC approximation and CPW approximation)

The decision criteria I implemented for testing were:

- Robertie
- Thorp (with Robertie's enhancement)
- Kleinman metric
- Chabot
- Trice (based on point of last take, i. e., Rule 62, or based on EPCs, using the "hidden" criterion given in his book)
- Nack-57
- Nack-58
- Keith (based on point of last take or based on CPW, using the percentages given in his article)
- Isight (based on point of last take or based on CPW)

Table 9: *Combinations and error*

	Adjusted pip count						
Decision criterion	Straight	Thorp	Keeler	Ward	Keith	Isight	Matussek
Robertie	9376	4753	3422	2839	3943	2856	
Thorp	7784	3438	2298	1838	2360	1709	
Kleinman	8301	4026	2466	1928	3031	1865	
Chabot	7819	2745	2118	1760	1518	1369	
Trice (Rule 62)	7954	3527	2143	1702	2016	1424	
Nack-57	7944	3503	2133	1684	1988	1415	
Nack-58	7948	3558	2140	1687	2007	1417	
Keith	7529	2861	1988	1835	1262	1233	
Isight	7228	2749	1743	1438	1474	1064	1934[a]
Keith (CPW)						1113[b]	2199[c]
Trice (EPC)						1909[d]	1498[e]

a. Matussek's CPW approximation, then Isight's percentages

b. Isight's CPW approximation, then Tom Keith's percentages

c. Matussek's CPW approximation, then Tom Keith's percentages

d. Isight's EPC approximation, then Walter Trice's EPC-based criterion

e. Matussek's EPC approximation, then Walter Trice's EPC-based criterion

Having all these building blocks available, it was very easy to test, e. g., which adjustments are best suited if you want to use the Kleinman metric as your decision criterion. I also checked combinations with Walter Trice's Rule 62, Nack Ballard's "Nack-57" and "Nack-58" (co-authored by Neil Kazaross), and a fairly new method by Michelin Chabot. The total cube error for various combinations is shown in **table 9**.

From the data, a couple of things are apparent: A straight pip count is much worse than the Thorp method, which in turn is worse than the Keeler method, which in turn is worse than the Ward method. This is just an extension of the work Tom Keith did in his investigation. His own adjusted pip count is typically worse than the Ward method, except when combined with his own decision criterion (which is of course the thing to do) or the very similar one by Michelin Chabot. The Ward method in turn is worse than the Isight method, except when combined with Bill Robertie's decision criterion.

You can also see that neither the metric by Danny Kleinman, nor Walter Trice's Rule 62 in combination with Jeff Ward's adjustments, nor Nack Ballard's methods, nor Michelin Chabot's method come close to what Tom Keith had achieved already 10 years ago. The same holds for the alternative approach of approximating EPCs and using an appropriate decision criterion suited for them.

Almost all decision criteria benefit when they are combined with Isight's adjusted pip count. And almost all adjusted pip counts benefit when combined with Isight's decision criterion. The best choice (in terms of minimum total error) is to combine Isight's adjusted pip count with Isight's decision criterion. Since Isight optimized both things simultaneously, this is a strong rationale for matching adjusted pip count and decision criterion.

It is also interesting to compare Walter Trice's Rule 62 with Nack-57/58 and the Chabot method: These decision criteria are all related to the "gold standard table" (more on this in the next section), which is reproduced exactly by Nack-57/58, but only approximated by Rule 62. Michelin Chabot in turn only approximates Rule 62 (by using a denominator of 8, no shift for the point of last take, and omitting the long/short race distinction), so one would expect to see the highest total error for Chabot, followed by Trice and Nack-57/58. However, for most of the adjusted pip counts, Chabot fares best of these four decision criteria. Why?

A possible explanation is that the "gold standard table" is valid only for low-wastage positions. However, like in the database of endgames played on FIBS, positions frequently show considerable wastage. This of course calls for adjusted pip counts, so the poor performance even of the "exact" Nack-57/58 is only to be expected. Getting the necessary adjustments right and matching them with an *adequate* decision criterion is more important than trying to squeeze out the last bit of equity using square roots and a distinction between long and short races. We will deal with the latter topic next.

A.3 Long Race or Short Race?

Why should one care about the distinction between long and short races (which introduces 3 additional parameters, i. e., the breakpoint between long and short races, the "short race denominator", and the "short race shift")? Well, Walter Trice came up with a table for money cube action in low-wastage positions. He was an expert in racing theory, so his table has been termed "gold standard table". It contains the maximum pip deficit for the non-roller (point of last take) depending on the roller's pip count. The corresponding graph is plotted in **figure 13** on the next page.

Walter Trice's "Rule 62" is able to generate this table (at least approximately) from a couple of simple arithmetical rules. Nack Ballard devised a new arithmetical rule, "Nack-57". Either this or its modification "Nack-58" manage to reproduce the table exactly (the "winner" is not yet clear

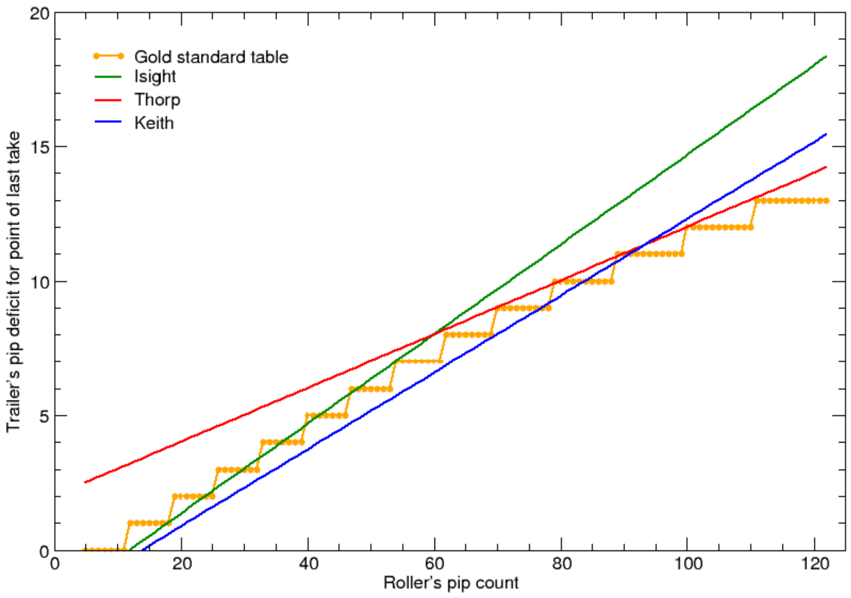


Figure 13: *"Gold standard table" approximations with straight lines*

Table 10: *Optimized parameters for decision criteria*

Denominator	Shift	Total error
3	−8	3034
4	−4	1778
5	−3	1145
6	−2	1064
7	−1	1144
8	−1	1329
9	0	1412
10	0	1597
11	1	1750
12	1	1860
13	1	2002
14	1	2161
15	1	2321
16	2	2403

from the rollouts), see `http://www.bgonline.org/forums/webbbs_config.pl?noframes;read=59714`. Both methods contain a distinction between long and short races with the breakpoints being, you guessed it, 62 and 57, respectively. You can see from figure 13 on the preceding page that methods without this distinction give a poorer fit to the data, e. g., the old THORP method (without BILL ROBERTIE's enhancement) with its long race denominator of 10 (which corresponds to a slope of 1/10) matches well only for the higher pip counts, while Isight's method with its long race denominator of 6 matches the lower pip counts well. TOM KEITH's method with its long race denominator of 7 is a compromise and does a good job overall. This looked like just another optimization problem well suited for an investigation with Isight: For a simple decision criterion without distinction between long and short races, **table 10** on the previous page lists the best shift value for each denominator along with the resulting total cube error. The lowest total error of 1064 is achieved with a denominator of 6 and a shift of −2, corresponding to the Isight method already presented in section 5.2 on page 16.

But if the Isight method is a rather poor fit of the "gold standard table" for the higher pip counts, three questions arise:

1. Why does it perform so well in comparison with other methods?
2. Why does it perform better than even the exact representation of the "gold standard table" by Nack-57/58?
3. Could it be improved even further by reintroducing the distinction between long and short races?

First, it performs so well because most of the endgames are rather short, see the distribution of the race length for TOM KEITH's database in **figure 14** on the next page: About 50 % of the races are shorter than 40 pips, 90 % are shorter than 70 pips, and 95 % are shorter than 75 pips. So fitting the straight line approximating the "gold standard table" to the shorter races pays in terms of increased accuracy. It is a number's game, so better adapt your heuristics to situations occuring frequently than to rather rare cases.

Second, Isight's decision criterion works better than Nack-57/58 for two reasons: The "gold standard table", as already mentioned, refers to *low-wastage positions*. If we define a low-wastage position as a position that does not require *any* adjustments by the Isight method, then TOM KEITH's database of around 50000 endgame positions contains only about 3000 of those (around 6 %). A "gold standard table" that cannot be reasonably applied to positions *with* wastage loses a lot of its value. The other reason is that decision criterion and adjusted pip count need to be matched: Imagine two different

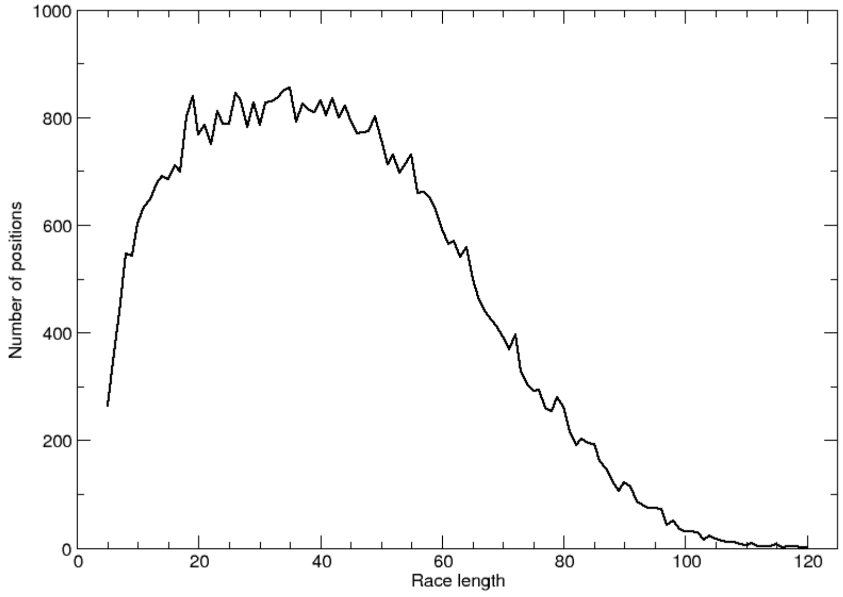


Figure 14: *Race length and positions*

methods for adjusting pip counts, one adding 100 pips per checker still on the board, while the other works with a straight pip count, decreased further by applying a bonus for a smooth position. It is obvious that these two methods must be paired with quite different decision criteria in order to give accurate cube decisions. Isight (and probably Tom Keith, too) optimized both adjusted pip count *and* decision criterion simultaneously.

Third: Yes, but only slightly. With 50 pips as the breakpoint, 5 as the short race denominator, −3 as the short race shift, 10 as the long race denominator, and 2 as the long race shift, you will get a total error of 1041, in comparison with the original 1064, at the cost of 3 additional parameters (and numbers to memorize). Applying the transformation formula then gives

$$p = \begin{cases} 84 - \frac{2}{5}l + 2\Delta l, & l \leq 50 \\ 74 - \frac{1}{5}l + 2\Delta l, & l \geq 50 \end{cases} \tag{15}$$

as the corresponding CPW-based decision criterion with a (re)doubling window of 70 %, 72 %, and 78 %, to be adapted in match play. To me, this method is not worth the effort, so it is mentioned just for reference. You decide.